`CSE-4029`| ADA Lab Assignment - *8* | `L25-L26`

Academic year : *2021-2022* | **Semester :** ***WIN***

Faculty Name : ***Dr. BKSP Kumarraju Alluri*** | **Date :** *28/5/2022*

Student Name : ***Taran Mamidala*** | **Reg. no. :** *19BCE7346*

Using Real Time Employee Dataset

Problem Statement

Given that, Understanding why and when employees are most likely to leave can lead to actions to improve employee retention as well as possibly planning new hiring in advance.

In this lab assignment we study a real time employee datset and solve the following problem statements :

> What is the likelihood of an active employee leaving the company?
> What are the key indicators of an employee leaving the company?
> What policies or strategies can be adopted based on the results to improve employee retention? **

Real-time Dataset Analysis

Importing libraries

In [ ]:

```
# importing libraries for data handling and analysis
import pandas as pd
from pandas.plotting import scatter_matrix
from pandas import ExcelWriter
from pandas import ExcelFile
```

```
from openpyxl import load_workbook
import numpy as np
from scipy.stats import norm, skew
from scipy import stats
import statsmodels.api as sm
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_tes
ting.py:19: FutureWarning: pandas.util.testing is deprecated.
Use the functions in the public API at pandas.testing instea
d.
  import pandas.util.testing as tm
```

In [ ]:

```
# importing libraries for data visualisations
import seaborn as sns
from matplotlib import pyplot
import matplotlib.pyplot as plt
import matplotlib.pylab as pylab
import matplotlib
%matplotlib inline
color = sns.color_palette()
from IPython.display import display
pd.options.display.max_columns = None
# Standard plotly imports
import plotly
# import plotly.plotly as py
import plotly.figure_factory as ff
import plotly.graph_objs as go
from plotly.offline import iplot, init_notebook_mode
```

In [ ]:

```
# importing miscellenous libraries
import os
import re
import sys
```

Importing the data

In [ ]:

```
# Read Excel file
df_original = pd.read_excel('/content/Real_emp_dataset.xlsx', sheet_name=0
)
print("Shape of dataframe is: {}".format(df_original.shape))
```

```
Shape of dataframe is: (1048, 35)
```

In [ ]:

```
# Making a copy of the original dataframe
data = df_original.copy()
```

Data Exploratory Visualisations

Data overview

In [ ]:

```
# Dataset columns
data.columns
```

Out[ ]:

```
Index(['Sr.No', 'Curr Status', 'Exit Date', 'Exit Type', 'Tit
le', 'Gender',
       'Designation', 'Department', 'Sub Department', 'Leve
l', 'Grade',
       'Blood Group', 'Employee Age', 'Qualification', 'Emplo
yee Status',
       'Employee Joining Type', 'Date of Birth', 'Date of Joi
ning',
       'Date of Retirement', 'Joining Salary', 'Summary Exp',
'Calculated Exp',
       'In/HO Exp', 'Total Exp', 'Service Agreement', 'Exit N
otice Period',
       'Marital Status', 'Year of Passing', 'Gross Salary',
'NPS', 'MEALCARD',
       'Total', 'CTC', 'FFC', 'Car Allowance'],
      dtype='object')
```

In [ ]:

```
str_var_list, num_var_list, all_var_list = get_dtypes(data=data)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:
31: DeprecationWarning:

`np.int` is a deprecated alias for the builtin `int`. To sile
nce this warning, use `int` by itself. Doing this will not mo
dify any behavior and is safe. When replacing `np.int`, you m
ay wish to use e.g. `np.int64` or `np.int32` to specify the p
recision. If you wish to review your current use, check the r
elease note link for additional information.
Deprecated in NumPy 1.20; for more details and guidance: http
s://numpy.org/devdocs/release/1.20.0-notes.html#deprecations

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:
31: DeprecationWarning:

`np.float` is a deprecated alias for the builtin `float`. To
silence this warning, use `float` by itself. Doing this will
not modify any behavior and is safe. If you specifically want
ed the numpy scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: http
s://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
```

In [ ]:

```
print(str_var_list) # string type
print(num_var_list) # numeric type
print(all_var_list) # all
```

```
['Exit Type', 'Title', 'Gender', 'Designation', 'Department',
'Sub Department', 'Level', 'Grade', 'Employee Age', 'Qualific
ation', 'Employee Status', 'Date of Birth', 'Date of Joinin
g', 'Date of Retirement', 'Summary Exp', 'Calculated Exp', 'I
n/HO Exp', 'Total Exp', 'Service Agreement', 'Marital Statu
s']
['Sr.No', 'Exit Date', 'Joining Salary', 'Exit Notice Perio
d', 'Year of Passing', 'Gross Salary', 'NPS', 'MEALCARD', 'To
tal', 'CTC']
['Exit Type', 'Title', 'Gender', 'Designation', 'Department',
'Sub Department', 'Level', 'Grade', 'Employee Age', 'Qualific
ation', 'Employee Status', 'Date of Birth', 'Date of Joinin
g', 'Date of Retirement', 'Summary Exp', 'Calculated Exp', 'I
n/HO Exp', 'Total Exp', 'Service Agreement', 'Marital Statu
s', 'Sr.No', 'Exit Date', 'Joining Salary', 'Exit Notice Peri
od', 'Year of Passing', 'Gross Salary', 'NPS', 'MEALCARD', 'T
otal', 'CTC']
```

In [ ]:

```
# Dataset header
data.head()
```

Out[ ]:

	Sr.No	Curr Status	Exit Date	Exit Type	Title	Gender	Designation	Department	S Departme
0	1	Active	NaT	NaN	Ms.	Female	Senior Manager	Project Management	Proj Manageme
1	2	Active	NaT	NaN	Mr.	Male	Senior Executive	FRD - Injectable	Formulatic Resear
2	3	Active	NaT	NaN	Ms.	Female	Deputy Manager	RAD- II	Regulat Affairs
3	4	Active	NaT	NaN	Ms.	Female	Senior Manager	Administration	Administrati
4	5	Active	NaT	NaN	Mr.	Male	Principal Scientist- II	FARD	Formulatic Analyti Resear

5 rows × 35 columns

> The dataset contains several numerical and categorical columns providing various information on employee's details.

In [ ]:

```
data.columns.to_series().groupby(data.dtypes).groups
```

Out[ ]:

```
{int64: ['Sr.No', 'CTC'], datetime64[ns]: ['Exit Date', 'Date
of Birth', 'Date of Joining', 'Date of Retirement'], float64:
['Joining Salary', 'Exit Notice Period', 'Year of Passing',
```

```
'Gross Salary', 'NPS', 'MEALCARD', 'Total', 'FFC', 'Car Allow
ance'], object: ['Curr Status', 'Exit Type', 'Title', 'Gende
r', 'Designation', 'Department', 'Sub Department', 'Level',
'Grade', 'Blood Group', 'Employee Age', 'Qualification', 'Emp
loyee Status', 'Employee Joining Type', 'Summary Exp', 'Calcu
lated Exp', 'In/HO Exp', 'Total Exp', 'Service Agreement', 'M
arital Status']}
```

In [ ]:

```
# Columns datatypes and missing values
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048 entries, 0 to 1047
Data columns (total 35 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Sr.No                  1048 non-null   int64
 1   Curr Status            1048 non-null   object
 2   Exit Date              410 non-null    datetime64[ns]
 3   Exit Type              410 non-null    object
 4   Title                  1048 non-null   object
 5   Gender                 1048 non-null   object
 6   Designation            1048 non-null   object
 7   Department             1048 non-null   object
 8   Sub Department         1048 non-null   object
 9   Level                  1048 non-null   object
 10  Grade                  856 non-null    object
 11  Blood Group            1022 non-null   object
 12  Employee Age           1046 non-null   object
 13  Qualification          1048 non-null   object
 14  Employee Status        1048 non-null   object
 15  Employee Joining Type  746 non-null    object
 16  Date of Birth          1047 non-null   datetime64[ns]
 17  Date of Joining        1048 non-null   datetime64[ns]
 18  Date of Retirement     1048 non-null   datetime64[ns]
 19  Joining Salary         1045 non-null   float64
 20  Summary Exp            641 non-null    object
 21  Calculated Exp         1048 non-null   object
 22  In/HO Exp              1048 non-null   object
 23  Total Exp              1048 non-null   object
 24  Service Agreement      1048 non-null   object
 25  Exit Notice Period     955 non-null    float64
 26  Marital Status         942 non-null    object
 27  Year of Passing        439 non-null    float64
 28  Gross Salary           1045 non-null   float64
 29  NPS                    566 non-null    float64
 30  MEALCARD               1035 non-null   float64
 31  Total                  1042 non-null   float64
 32  CTC                    1048 non-null   int64
 33  FFC                    566 non-null    float64
 34  Car Allowance          562 non-null    float64
dtypes: datetime64[ns](4), float64(9), int64(2), object(20)
memory usage: 286.7+ KB
```

Data Visulaization

In [ ]:

```
# Replacing the 'current status' column with integers before performing an
y visualizations
data['Curr Status'] = data['Curr Status'].apply(lambda x: 1 if x == 'Activ
e' else 0)
```

In [ ]:

```
data.head(4)
```

Out[ ]:

	Sr.No	Curr Status	Exit Date	Exit Type	Title	Gender	Designation	Department	S Departme
0	1	1	NaT	NaN	Ms.	Female	Senior Manager	Project Management	Proj Manageme
1	2	1	NaT	NaN	Mr.	Male	Senior Executive	FRD - Injectable	Formulatic Resear
2	3	1	NaT	NaN	Ms.	Female	Deputy Manager	RAD- II	Regulat Affairs
3	4	1	NaT	NaN	Ms.	Female	Senior Manager	Administration	Administrati

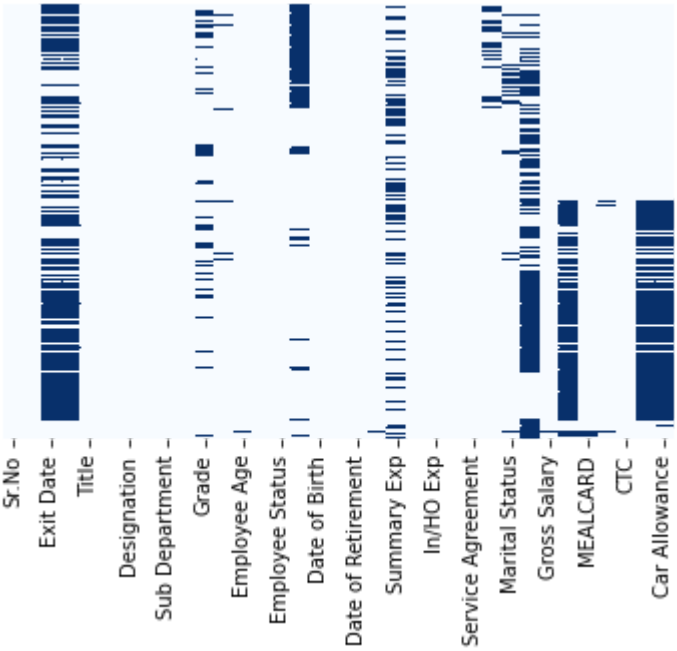
4 rows × 35 columns

In [ ]:

```
# Let's see if we have any missing data
sns.heatmap(data.isnull(), yticklabels = False, cbar = False, cmap="Blues"
)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4376051450>
```

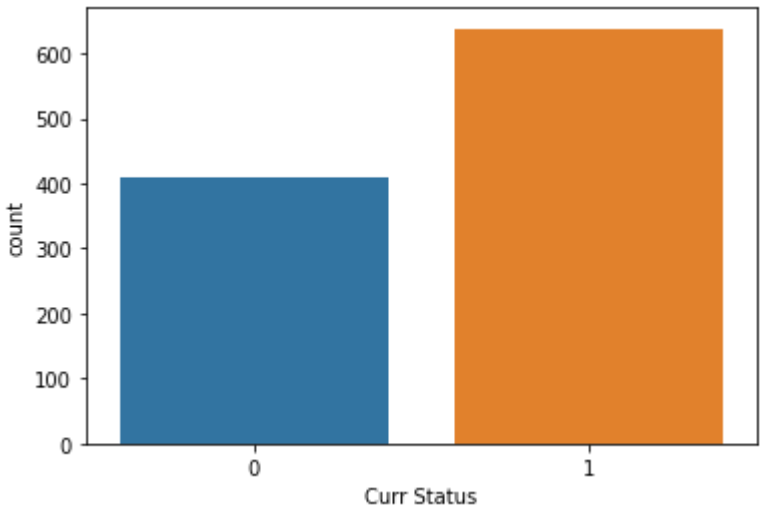


Dealing with Missing Values

In [ ]:

```
sns.countplot('Curr Status', data=data);
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.p
y:43: FutureWarning: Pass the following variable as a keyword
arg: x. From version 0.12, the only valid positional argument
will be `data`, and passing other arguments without an explic
it keyword will result in an error or misinterpretation.
  FutureWarning
```



In [ ]:

```
data.isnull().any().any()
```

Out[ ]:

```
True
```

Listwise deletion

excluding all cases (listwise) that have missing values

In [ ]:

```
import pandas as pd
import numpy as np
import seaborn as sns

import os
plt.style.use('seaborn-colorblind')
%matplotlib inline
```

In [ ]:

```
m = check_missing(data=data,output_path=r'/content/')
m
```

```
result saved at /content/ missing.csv
```

Out[ ]:

	total missing	proportion
Sr.No	0	0.000000
Exit Date	0	0.000000
Exit Type	638	0.608779
Title	0	0.000000
Gender	0	0.000000
Designation	0	0.000000
Department	0	0.000000
Sub Department	0	0.000000
Level	0	0.000000
Grade	192	0.183206
Employee Age	2	0.001908
Qualification	0	0.000000
Employee Status	0	0.000000
Date of Birth	1	0.000954
Date of Joining	0	0.000000
Date of Retirement	0	0.000000
Joining Salary	3	0.002863
Summary Exp	407	0.388359
Calculated Exp	0	0.000000
In/HO Exp	0	0.000000
Total Exp	0	0.000000
Service Agreement	0	0.000000
Exit Notice Period	93	0.088740
Marital Status	106	0.101145
Year of Passing	609	0.581107
Gross Salary	3	0.002863
NPS	482	0.459924
MEALCARD	13	0.012405
Total	6	0.005725
CTC	0	0.000000

Adding a variable to denote NA

creating an additional variable indicating whether the data was missing for that observation

In [ ]:

```
# extra_NA is created, 0-not missing 1-missing for that observation
data3 = add_var_denote_NA(data=data,NA_col=[ ])
data3
# print(data3.extra_is_NA.value_counts())
# data3.head(8)
```

Out[ ]:

	Sr.No	Curr Status	Exit Date	Exit Type	Title	Gender	Designation	Department
0	1	Active	NaT	NaN	Ms.	Female	Senior Manager	Project Management
1	2	Active	NaT	NaN	Mr.	Male	Senior Executive	FRD - Injectable
2	3	Active	NaT	NaN	Ms.	Female	Deputy Manager	RAD- II
3	4	Active	NaT	NaN	Ms.	Female	Senior Manager	Administration
4	5	Active	NaT	NaN	Mr.	Male	Principal Scientist- II	FARD
...	...	...	...	...	...	...	...	...
1043	1044	Inactive	2019-08-26	Resigned	Mr.	Male	Scientist - I	RAD- II
1044	1045	Inactive	2019-07-11	Absconding	Mr.	Male	Assistant	RAD- II
1045	1046	Inactive	2019-06-06	Resigned	Ms.	Female	Research Associate - I	RAD- II
1046	1047	Inactive	2019-07-11	Absconding	Ms.	Female	Research Associate - I	RAD- II
1047	1048	Inactive	2019-05-31	Absconding	Mr.	Male	Assistant	RAD- II

1048 rows × 35 columns

In [ ]:

```
# which has NA has been dropped
data2 = drop_missing(data=data)
data2.shape
```

In [ ]:

```
# Print the unique values of the "department" column
print(data.Department.unique())
print(data.Level.unique())
```

```
['Project Management' 'FRD -Injectable' 'RAD- II' 'Administra
tion' 'FARD'
 'FRD' 'FARD - Injectable' 'CPD-BIO' 'HRD' 'CPD-PK' 'CPD-QA'
'FRD-QA'
 'Stores' 'Packaging & Development' 'Corporate Stability' 'H
R'
 'CPD-BIO STATISTICS' 'FRD - Ophthalmic' 'Purchase'
 'Supply Chain Management' 'Information Systems' 'EHS' 'CPD'
 'FRD - INJECTABLE']
['Middle Management' 'Executive' 'Junior Management' 'Senior
Management'
 'Assistant']
```

In [ ]:

```
# It makes sense to drop 'Blood Group' , 'Date of Birth' and 'Exit Notice
 Period' since they do not change from one employee to the other
# Let's drop 'Car Allowance' as well
data.drop(['Blood Group', 'Date of Birth', 'Exit Notice Period', 'Car Allo
wance'], axis=1, inplace=True)
```

In [ ]:

```
# Let's see how many employees left the company!
left_employees    = data[data['Curr Status'] == 0]
stayed_employees  = data[data['Curr Status'] == 1]
```

In [ ]:

```
# Count the number of employees who stayed and left
# It seems that we are dealing with an imbalanced dataset

print("Total =", len(data))

print("Number of employees who left the company =", len(left_employees))
print("Percentage of employees who left the company =", 1.*len(left_employ
ees)/len(data)*100.0, "%")

print("Number of employees who did not leave the company (stayed) =", len(
stayed_employees))
print("Percentage of employees who did not leave the company (stayed) =",
1.*len(stayed_employees)/len(data)*100.0, "%")
```

```
Total = 1048
Number of employees who left the company = 410
Percentage of employees who left the company = 39.12213740458
015 %
Number of employees who did not leave the company (stayed) =
638
Percentage of employees who did not leave the company (staye
d) = 60.87786259541985 %
```

In [ ]:

```
# Let's see if we have any missing data
sns.heatmap(data.isnull(), yticklabels = False, cbar = False, cmap="Blues"
)
```

Numerical features overview

In [ ]:

```
data.describe()
```

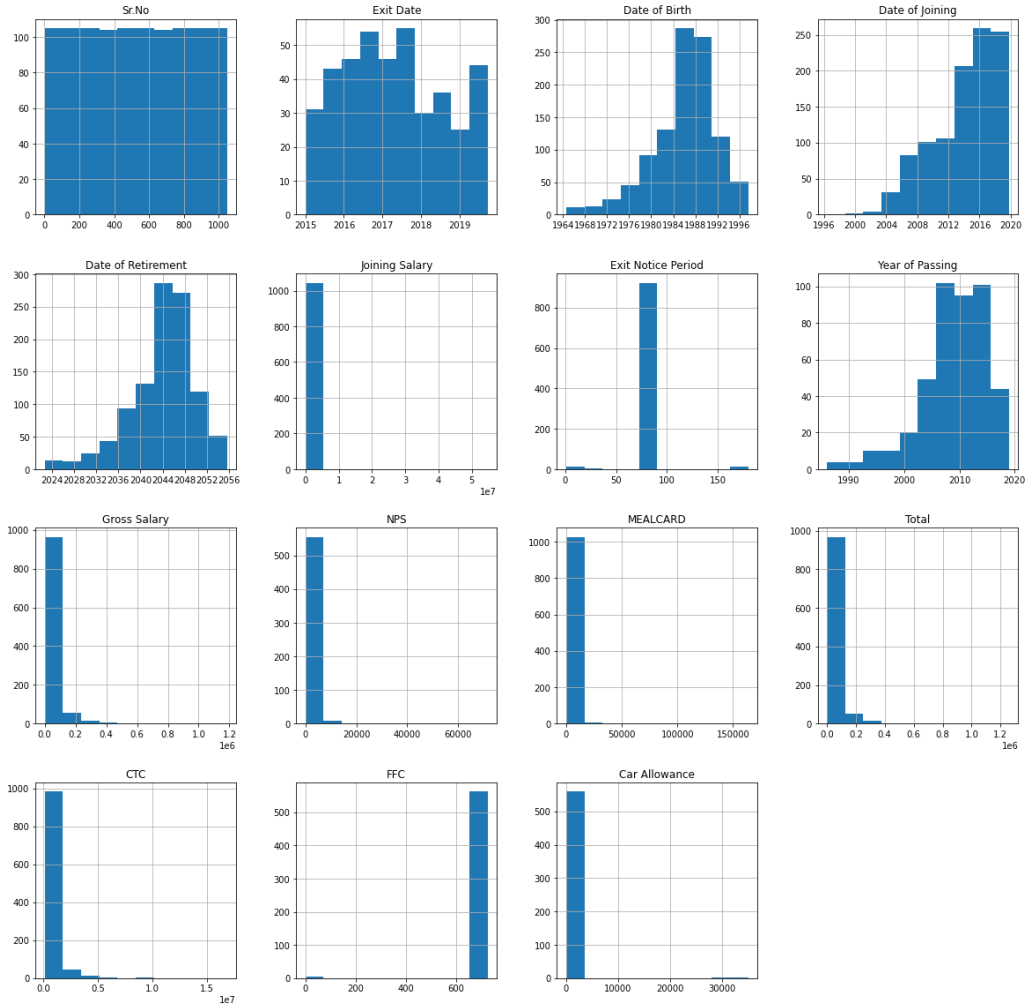
Out[ ]:

	Sr.No	Curr Status	Joining Salary	Exit Notice Period	Year of Passing	Gross Sa
count	1048.000000	1048.000000	1.045000e+03	955.000000	439.000000	1.045000e

mean	524.500000	0.608779	2.107968e+05	89.788482	2009.314351	5.840510e
std	302.675844	0.488257	1.728889e+06	15.675806	5.976862	6.941642e
min	1.000000	0.000000	0.000000e+00	1.000000	1986.000000	0.000000e
25%	262.750000	0.000000	1.200000e+04	90.000000	2006.000000	2.753100e
50%	524.500000	1.000000	2.633800e+04	90.000000	2010.000000	4.247100e
75%	786.250000	1.000000	1.478640e+05	90.000000	2014.000000	6.730400e
max	1048.000000	1.000000	5.475280e+07	180.000000	2019.000000	1.184348e

In [ ]:

```
data.hist(figsize=(20,20))
plt.show()
```



A few observations can be made based on the information and histograms for numerical features:

- Many histograms are tail-heavy; indeed several distributions are right-skewed (e.g. MonthlyIncome DistanceFromHome, YearsAtCompany). Data transformation methods may be required to approach a normal distribution prior to fitting a model to the data.
- Age distribution is a slightly right-skewed normal distribution with the bulk of the staff between 25 and 45 years old.

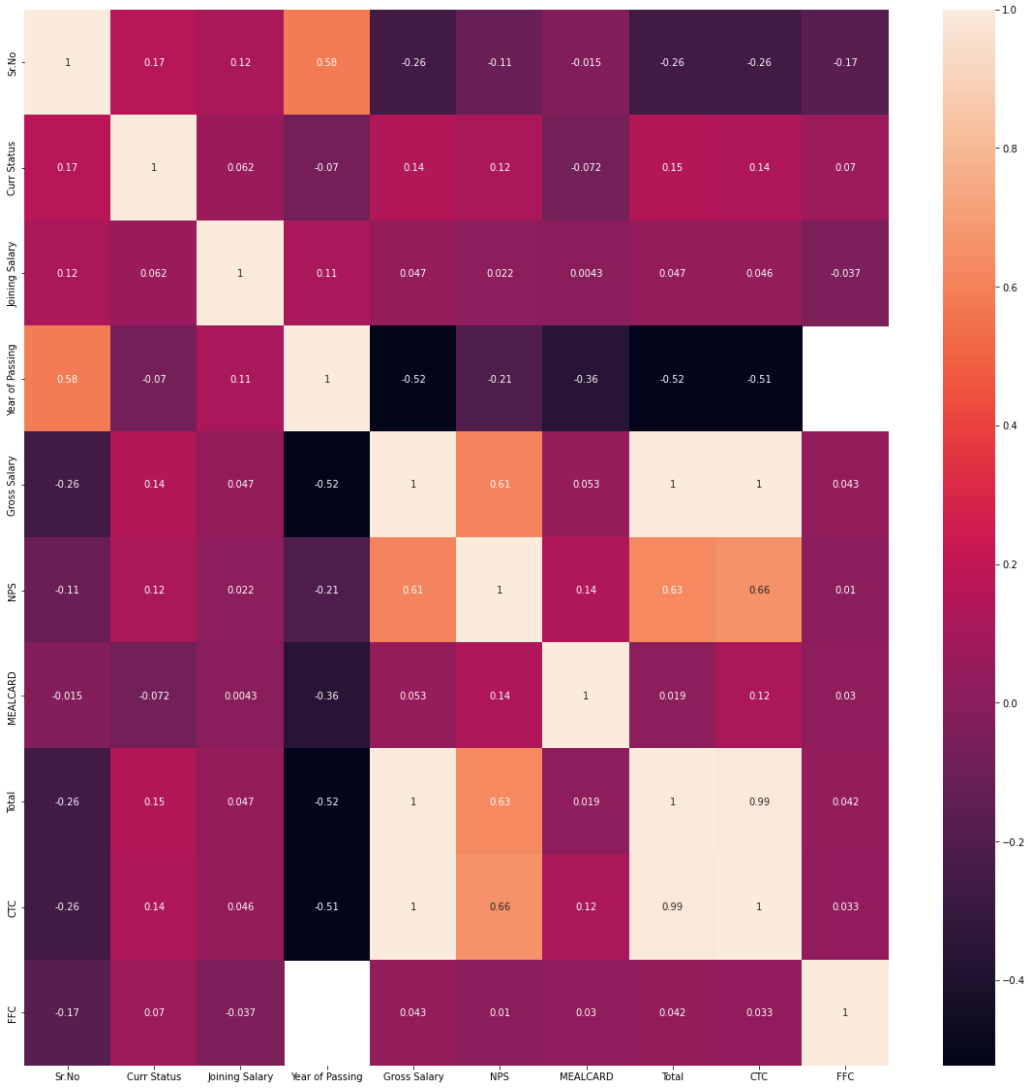
- EmployeeCount and StandardHours are constant values for all employees. They're likely to be redundant features.
- Employee Number is likely to be a unique identifier for employees given the feature's quasi-uniform distribution.

In [ ]:

```
correlations = data.corr()
f, ax = plt.subplots(figsize = (20, 20))
sns.heatmap(correlations, annot = True)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f606f62f4d0>
```

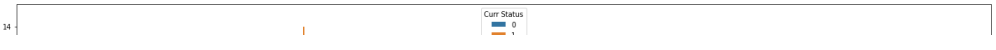


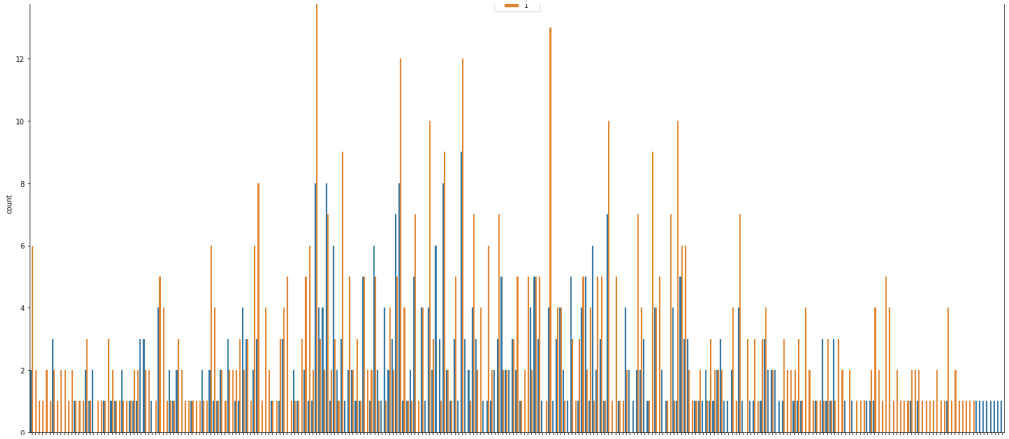
In [ ]:

```
plt.figure(figsize=[25, 12])
sns.countplot(x = 'Employee Age', hue = 'Curr Status', data = data)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f606c7e00d0>
```





Feature distribution by target attribute

Employee Age

> The age distributions for Active and Ex-employees

In [ ]:

```
(mu, sigma) = norm.fit(data.loc[data['Curr Status'] == '0', 'Employee Age'
])
print('Ex-exmployees: average age = {:.1f} years old and standard deviatio
n = {:.1f}'.format(mu, sigma))
(mu, sigma) = norm.fit(data.loc[data['Curr Status'] == '1', 'Employee Age'
])
print('Current exmployees: average age = {:.1f} years old and standard dev
iation = {:.1f}'.format(mu, sigma))
```

> kernel density estimation (KDE) plot colored by the value of the target.

In [ ]:

```
# Add histogram data
x1 = data.loc[data['Curr Status'] == '1', 'Employee Age']
x2 = data.loc[data['Curr Status'] == '0', 'Employee Age']
# Group data together
hist_data = [x1, x2]
group_labels = ['Active Employees', 'Ex-Employees']
# Create distplot with custom bin_size
fig = ff.create_distplot(hist_data, group_labels, bin_size=10, curve_type=
'kde', show_hist='TRUE')
# Add title
fig['layout'].update(title='Employee Age Distribution in Percent by Curr S
tatus Status')
fig['layout'].update(xaxis=dict(range=[15, 60], dtick=5))
# Plot
py.iplot(fig, filename='Distplot with Multiple Datasets')
```

Gender

> Gender distribution shows that the dataset features a higher relative proportion of male ex-employees than female ex-employees, with normalised gender distribution of ex-employees in the dataset at 17.0% for Males and 14.8% for Females.

In [ ]:

```
from pandas.core.frame import DataFrame
# Gender of employees
data['Gender'].value_counts()
```

Out[ ]:

```
Male      809
Female    239
Name: Gender, dtype: int64
```

In [ ]:

```
print("Normalised gender distribution of ex-employees in the dataset: Male
= {:.1f}%; Female {:.1f}%.".format((data[(data['Curr Status'] == '0') & (
    data['Gender'] == 'Male')].shape[0] / data[data['Gender'] == 'Male'].s
hape[0])*100, (data[(data['Curr Status'] == '0') & (data['Gender'] == 'Fem
ale')].shape[0] / data[data['Gender'] == 'Female'].shape[0])*100))
```

```
Normalised gender distribution of ex-employees in the datase
t: Male = 0.0%; Female 0.0%.
```

In [ ]:

```
df_Gender = pd.DataFrame(columns=["Gender", "Levels"])
i=0
for field in list(data['Gender'].unique()):
    ratio = data[(data['Gender']==field)&(data['Curr Status']=="0")].shape
[0] / data[data['Gender']==field].shape[0]*100
    df_Gender.loc[i] = (field, ratio*100)
    i += 1

df_G = df_Gender.groupby(by="Gender").sum()
df_G.iplot(kind='bar',title='Levels by Gender (%)')
```

Department

> The data features employee data from three departments: Research & Development, Sales, and Human Resources.

In [ ]:

```
# The organisation consists of several departments
data['Department'].value_counts()
```

Out[ ]:

```
RAD- II                       361
FARD                          270
FRD                           130
FARD - Injectable             101
FRD -Injectable                58
Project Management             32
Purchase                       17
Packaging & Development        17
CPD-PK                         11
HR                              9
CPD-BIO                         7
Administration                  6
CPD                             6
CPD-QA                          6
Stores                          3
Corporate Stability             3
Information Systems             2
HRD                             2
FRD - INJECTABLE                2
CPD-BIO STATISTICS              1
FRD - Ophthalmic                1
FRD-QA                          1
Supply Chain Management         1
EHS                             1
Name: Department, dtype: int64
```

In [ ]:

```
df_Department = pd.DataFrame(columns=["Department", "Level"])
i=0
for field in list(data['Department'].unique()):
    ratio = data[(data['Department']==field)&(data['Curr Status']=="0")].s
hape[0] / data[data['Department']==field].shape[0]
    df_Department.loc[i] = (field, ratio*100)
    i += 1

df_DF = df_Department.groupby(by="Department").sum()
df_DF.iplot(kind='bar',title='Leavers by Department (%)')
```

Designation

> A preliminary look at the relationship between Business Travel frequency and Attrition Status shows that there is a largest normalized proportion of Leavers for employees that travel "frequently". Travel metrics associated with Business Travel status were not disclosed (i.e. how many hours of Travel is considered "Frequent").

In [ ]:

```
# Employees have different business travel commitmnent depending on their
 roles and level in the organisation
df_HR['BusinessTravel'].value_counts()
```

Out[ ]:

```
Travel_Rarely        1043
Travel_Frequently     277
Non-Travel            150
Name: BusinessTravel, dtype: int64
```

In [ ]:

```
df_BusinessTravel = pd.DataFrame(columns=["Business Travel", "% of Leaver
s"])
i=0
for field in list(df_HR['BusinessTravel'].unique()):
    ratio = df_HR[(df_HR['BusinessTravel']==field)&(df_HR['Attrition']=="Y
es")].shape[0] / df_HR[df_HR['BusinessTravel']==field].shape[0]
    df_BusinessTravel.loc[i] = (field, ratio*100)
    i += 1
    #print("In {}, the ratio of leavers is {:.2f}%".format(field, ratio*10
0))
df_BT = df_BusinessTravel.groupby(by="Business Travel").sum()
df_BT.iplot(kind='bar',title='Leavers by Business Travel (%)')
```

> Several Job Roles are listed in the dataset: Sales Executive, Research Scientist, Laboratory Technician, Manufacturing Director, Healthcare Representative, Manager, Sales Representative, Research Director, Human Resources.

In [ ]:

```
# Employees in the database have several roles on-file
df_HR['JobRole'].value_counts()
```

Out[ ]:

```
Sales Executive              326
Research Scientist           292
Laboratory Technician        259
Manufacturing Director       145
Healthcare Representative    131
Manager                      102
Sales Representative          83
Research Director             80
Human Resources               52
Name: JobRole, dtype: int64
```

In [ ]:

```
df_JobRole = pd.DataFrame(columns=["Job Role", "% of Leavers"])
i=0
for field in list(df_HR['JobRole'].unique()):
    ratio = df_HR[(df_HR['JobRole']==field)&(df_HR['Attrition']=="Yes")].s
hape[0] / df_HR[df_HR['JobRole']==field].shape[0]
    df_JobRole.loc[i] = (field, ratio*100)
    i += 1
    #print("In {}, the ratio of leavers is {:.2f}%".format(field, ratio*10
0))
```

```
df_JR = df_JobRole.groupby(by="Job Role").sum()
df_JR.iplot(kind='bar',title='Leavers by Job Role (%)')
```

> Employees have an assigned level within the organisation which varies from 1 (staff) to 5 (managerial/director). Employees with an assigned Job Level of "1" show the largest normalized proportion of Leavers.

In [ ]:

```
df_HR['JobLevel'].value_counts()
```

Out[ ]:

```
1    543
2    534
3    218
4    106
5     69
Name: JobLevel, dtype: int64
```

In [ ]:

```
df_JobLevel = pd.DataFrame(columns=["Job Level", "% of Leavers"])
i=0
for field in list(df_HR['JobLevel'].unique()):
```

```
    ratio = df_HR[(df_HR['JobLevel']==field)&(df_HR['Attrition']=="Yes")].
shape[0] / df_HR[df_HR['JobLevel']==field].shape[0]
    df_JobLevel.loc[i] = (field, ratio*100)
    i += 1
    #print("In {}, the ratio of leavers is {:.2f}%".format(field, ratio*10
0))
df_JL = df_JobLevel.groupby(by="Job Level").sum()
df_JL.iplot(kind='bar',title='Leavers by Job Level (%)')
```

> A ranking is associated to the employee's Job Involvement :1 'Low' 2 'Medium' 3 'High' 4 'Very High'. The plot below indicates a negative correlation with the Job Involvement of an employee and the Attrition Status. In other words, employees with higher Job Involvement are less likely to leave.

In [ ]:

```
df_HR['JobInvolvement'].value_counts()
```

Out[ ]:

```
3    868
2    375
4    144
1     83
Name: JobInvolvement, dtype: int64
```

In [ ]:

```
df_JobInvolvement = pd.DataFrame(columns=["Job Involvement", "% of Leaver
s"])
i=0
for field in list(df_HR['JobInvolvement'].unique()):
    ratio = df_HR[(df_HR['JobInvolvement']==field)&(df_HR['Attrition']=="Y
es")].shape[0] / df_HR[df_HR['JobInvolvement']==field].shape[0]
    df_JobInvolvement.loc[i] = (field, ratio*100)
    i += 1
    #print("In {}, the ratio of leavers is {:.2f}%".format(field, ratio*10
0))
df_JI = df_JobInvolvement.groupby(by="Job Involvement").sum()
df_JI.iplot(kind='bar',title='Leavers by Job Involvement (%)')
```

> The data indicates that employees may have access to some Training. A feature indicates how many years it's been since the employee attended such training.

In [ ]:

```
print("Number of training times last year varies from {} to {} years.".for
mat(
    df_HR['TrainingTimesLastYear'].min(), df_HR['TrainingTimesLastYear'].m
ax()))
```

Number of training times last year varies from 0 to 6 years.

In [ ]:

```
# Add histogram data
x1 = df_HR.loc[df_HR['Attrition'] == 'No', 'TrainingTimesLastYear']
x2 = df_HR.loc[df_HR['Attrition'] == 'Yes', 'TrainingTimesLastYear']
# Group data together
hist_data = [x1, x2]
group_labels = ['Active Employees', 'Ex-Employees']
# Create distplot with custom bin_size
fig = ff.create_distplot(hist_data, group_labels,
                         curve_type='kde', show_hist=False, show_rug=False
)
# Add title
fig['layout'].update(
    title='Training Times Last Year metric in Percent by Attrition Status'
)
fig['layout'].update(xaxis=dict(range=[0, 6], dtick=1))
# Plot
py.iplot(fig, filename='Distplot with Multiple Datasets')
```

Out[ ]:

> There is a feature for the number of companies the employee has worked at.

> 0 likely indicates that according to records, the employee has only worked at this company

In [ ]:

```
df_HR['NumCompaniesWorked'].value_counts()
```

Out[ ]:

```
1    521
0    197
3    159
2    146
4    139
7     74
6     70
5     63
9     52
8     49
Name: NumCompaniesWorked, dtype: int64
```

In [ ]:

```
df_NumCompaniesWorked = pd.DataFrame(columns=["Num Companies Worked", "% o
f Leavers"])
i=0
for field in list(df_HR['NumCompaniesWorked'].unique()):
    ratio = df_HR[(df_HR['NumCompaniesWorked']==field)&(df_HR['Attrition']
=="Yes")].shape[0] / df_HR[df_HR['NumCompaniesWorked']==field].shape[0]
    df_NumCompaniesWorked.loc[i] = (field, ratio*100)
    i += 1
    #print("In {}, the ratio of leavers is {:.2f}%".format(field, ratio*10
0))
df_NC = df_NumCompaniesWorked.groupby(by="Num Companies Worked").sum()
df_NC.iplot(kind='bar',title='Leavers by Num Companies Worked (%)')
```

Employee Years at the Company

In [ ]:

```
df_HR
```

Out[ ]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome
0	41	Yes	Travel_Rarely	1102	Sales	1
1	49	No	Travel_Frequently	279	Research & Development	8
2	37	Yes	Travel_Rarely	1373	Research & Development	2
3	33	No	Travel_Frequently	1392	Research & Development	3
4	27	No	Travel_Rarely	591	Research & Development	2
5	32	No	Travel_Frequently	1005	Research & Development	2
6	59	No	Travel_Rarely	1324	Research & Development	3
7	30	No	Travel_Rarely	1358	Research & Development	24
8	38	No	Travel_Frequently	216	Research & Development	23
9	36	No	Travel_Rarely	1299	Research & Development	27
10	35	No	Travel_Rarely	809	Research & Development	16
11	29	No	Travel_Rarely	153	Research & Development	15
12	31	No	Travel_Rarely	670	Research & Development	26
13	34	No	Travel_Rarely	1346	Research & Development	19
14	28	Yes	Travel_Rarely	103	Research & Development	24
15	29	No	Travel_Rarely	1389	Research & Development	21
16	32	No	Travel_Rarely	334	Research & Development	5

17	22	No	Non-Travel	1123	Research & Development	16
18	53	No	Travel_Rarely	1219	Sales	2
19	38	No	Travel_Rarely	371	Research & Development	2
20	24	No	Non-Travel	673	Research & Development	11
21	36	Yes	Travel_Rarely	1218	Sales	9
22	34	No	Travel_Rarely	419	Research & Development	7
23	21	No	Travel_Rarely	391	Research & Development	15
24	34	Yes	Travel_Rarely	699	Research & Development	6
25	53	No	Travel_Rarely	1282	Research & Development	5
26	32	Yes	Travel_Frequently	1125	Research & Development	16
27	42	No	Travel_Rarely	691	Sales	8
28	44	No	Travel_Rarely	477	Research & Development	7
29	46	No	Travel_Rarely	705	Sales	2
...	...	...	...	...	...	...
1440	36	No	Travel_Frequently	688	Research & Development	4
1441	56	No	Non-Travel	667	Research & Development	1
1442	29	Yes	Travel_Rarely	1092	Research & Development	1
1443	42	No	Travel_Rarely	300	Research & Development	2
1444	56	Yes	Travel_Rarely	310	Research & Development	7
1445	41	No	Travel_Rarely	582	Research & Development	28
1446	34	No	Travel_Rarely	704	Sales	28
1447	36	No	Non-Travel	301	Sales	15
1448	41	No	Travel_Rarely	930	Sales	3
1449	32	No	Travel_Rarely	529	Research & Development	2
1450	35	No	Travel_Rarely	1146	Human Resources	26
1451	38	No	Travel_Rarely	345	Sales	10
1452	50	Yes	Travel_Frequently	878	Sales	1

1453	36	No	Travel_Rarely	1120	Sales	11
1454	45	No	Travel_Rarely	374	Sales	20
1455	40	No	Travel_Rarely	1322	Research & Development	2
1456	35	No	Travel_Frequently	1199	Research & Development	18
1457	40	No	Travel_Rarely	1194	Research & Development	2
1458	35	No	Travel_Rarely	287	Research & Development	1
1459	29	No	Travel_Rarely	1378	Research & Development	13
1460	29	No	Travel_Rarely	468	Research & Development	28
1461	50	Yes	Travel_Rarely	410	Sales	28
1462	39	No	Travel_Rarely	722	Sales	24
1463	31	No	Non-Travel	325	Research & Development	5
1464	26	No	Travel_Rarely	1167	Sales	5
1465	36	No	Travel_Frequently	884	Research & Development	23
1466	39	No	Travel_Rarely	613	Research & Development	6
1467	27	No	Travel_Rarely	155	Research & Development	4
1468	49	No	Travel_Frequently	1023	Sales	2
1469	34	No	Travel_Rarely	628	Research & Development	8

1470 rows × 35 columns

In [ ]:

```
print('Average Number of Years at the company for currently active employe
es: {:.2f} miles and ex-employees: {:.2f} years'.format(
    df_HR[df_HR['Attrition'] == 'No']['YearsAtCompany'].mean(), df_HR[df_H
R['Attrition'] == 'Yes']['YearsAtCompany'].mean()))
```

```
Average Number of Years at the company for currently active e
mployees: 7.37 miles and ex-employees: 5.13 years
```

In [ ]:

```
print("Number of Years at the company varies from {} to {} years.".format(
    df_HR['YearsAtCompany'].min(), df_HR['YearsAtCompany'].max()))
```

```
Number of Years at the company varies from 0 to 40 years.
```

In [ ]:

```
# Add histogram data
x1 = df_HR.loc[df_HR['Attrition'] == 'No', 'YearsAtCompany']
x2 = df_HR.loc[df_HR['Attrition'] == 'Yes', 'YearsAtCompany']
# Group data together
hist_data = [x1, x2]
group_labels = ['Active Employees', 'Ex-Employees']
# Create distplot with custom bin_size
fig = ff.create_distplot(hist_data, group_labels,
                         curve_type='kde', show_hist=False, show_rug=False
)
# Add title
fig['layout'].update(title='Years At Company in Percent by Attrition Statu
s')
fig['layout'].update(xaxis=dict(range=[0, 40], dtick=5))
# Plot
py.iplot(fig, filename='Distplot with Multiple Datasets')
```

Out[ ]:

In [ ]:

```
print("Number of Years in the current role varies from {} to {} years.".fo
rmat(
    df_HR['YearsInCurrentRole'].min(), df_HR['YearsInCurrentRole'].max()))
```

```
Number of Years in the current role varies from 0 to 18 year
```

s.

In [ ]:

```
# Add histogram data
x1 = df_HR.loc[df_HR['Attrition'] == 'No', 'YearsInCurrentRole']
x2 = df_HR.loc[df_HR['Attrition'] == 'Yes', 'YearsInCurrentRole']
# Group data together
hist_data = [x1, x2]
group_labels = ['Active Employees', 'Ex-Employees']
# Create distplot with custom bin_size
fig = ff.create_distplot(hist_data, group_labels,
                         curve_type='kde', show_hist=False, show_rug=False
)
# Add title
fig['layout'].update(title='Years InCurrent Role in Percent by Attrition S
tatus')
fig['layout'].update(xaxis=dict(range=[0, 18], dtick=1))
# Plot
py.iplot(fig, filename='Distplot with Multiple Datasets')
```

Out[ ]:

In [ ]:

```
print("Number of Years since last promotion varies from {} to {} years.".f
ormat(
```

```
    df_HR['YearsSinceLastPromotion'].min(), df_HR['YearsSinceLastPromotio
n'].max()))
```

```
Number of Years since last promotion varies from 0 to 15 year
s.
```

In [ ]:

```
# Add histogram data
x1 = df_HR.loc[df_HR['Attrition'] == 'No', 'YearsSinceLastPromotion']
x2 = df_HR.loc[df_HR['Attrition'] == 'Yes', 'YearsSinceLastPromotion']
# Group data together
hist_data = [x1, x2]
group_labels = ['Active Employees', 'Ex-Employees']
# Create distplot with custom bin_size
fig = ff.create_distplot(hist_data, group_labels,
                         curve_type='kde', show_hist=False, show_rug=False
)
# Add title
fig['layout'].update(title='Years Since Last Promotion in Percent by Attri
tion Status')
fig['layout'].update(xaxis=dict(range=[0, 15], dtick=1))
# Plot
py.iplot(fig, filename='Distplot with Multiple Datasets')
```

Out[ ]:

In [ ]:

```
print("Total working years varies from {} to {} years.".format(
    df_HR['TotalWorkingYears'].min(), df_HR['TotalWorkingYears'].max()))
```

```
Total working years varies from 0 to 40 years.
```

In [ ]:

```
# Add histogram data
x1 = df_HR.loc[df_HR['Attrition'] == 'No', 'TotalWorkingYears']
x2 = df_HR.loc[df_HR['Attrition'] == 'Yes', 'TotalWorkingYears']
# Group data together
hist_data = [x1, x2]
group_labels = ['Active Employees', 'Ex-Employees']
# Create distplot with custom bin_size
fig = ff.create_distplot(hist_data, group_labels,
                         curve_type='kde', show_hist=False, show_rug=False
)
# Add title
fig['layout'].update(title='Total Working Years in Percent by Attrition St
atus')
fig['layout'].update(xaxis=dict(range=[0, 40], dtick=5))
# Plot
py.iplot(fig, filename='Distplot with Multiple Datasets')
```

Out[ ]:

In [ ]:

```
# Add histogram data
x1 = data.loc[data['Curr Status'] == 'No', 'YearsWithCurrManager']
x2 = data.loc[data['Curr Status'] == 'Yes', 'YearsWithCurrManager']
# Group data together
hist_data = [x1, x2]
group_labels = ['Active Employees', 'Ex-Employees']
# Create distplot with custom bin_size
fig = ff.create_distplot(hist_data, group_labels,
                         curve_type='kde', show_hist=False, show_rug=False
)
# Add title
fig['layout'].update(
    title='Years With Curr Manager in Percent by Curr Status')
fig['layout'].update(xaxis=dict(range=[0, 17], dtick=1))
# Plot
py.iplot(fig, filename='Distplot with Multiple Datasets')
```

Out[ ]:

Target Variable: current state

In [ ]:

```
# Curr Status indicates if the employee is currently active ('No') or has
 left the company ('Yes')
data['Curr Status'].value_counts()
```

Out[ ]:

```
No     1233
Yes     237
Name: Attrition, dtype: int64
```

In [ ]:

```
print("Percentage of Current Employees is {:.1f}% and of Ex-employees is:
{:.1f}%".format(
    data[data['Curr Status'] == 'No'].shape[0] / data.shape[0]*100,
    data[data['Curr Status'] == 'Yes'].shape[0] / data.shape[0]*100))
```

```
Percentage of Current Employees is 83.9% and of Ex-employees
is: 16.1%
```

In [ ]:

```
data['Curr Status'].iplot(kind='hist', xTitle='Curr Status',
                          yTitle='count', title='Curr Status Distribution')
```

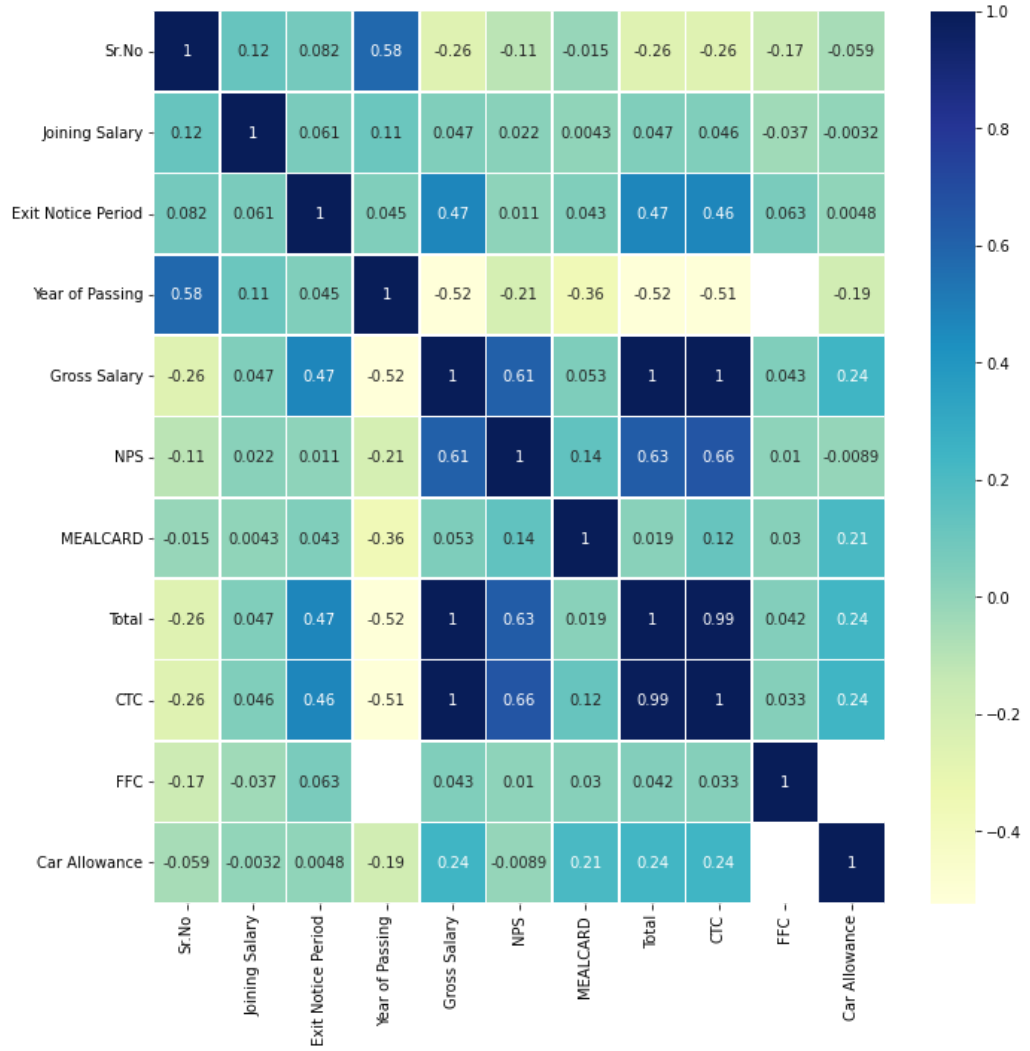
> As shown on the chart above, we see this is an imbalanced class problem. Indeed, the percentage of Current Employees in our dataset is 83.9% and the percentage of Ex-employees is: 16.1%
>
> Machine learning algorithms typically work best when the number of instances of each classes are roughly equal. We will have to address this target feature imbalance prior to implementing our Machine Learning algorithms.

Correlation

In [ ]:

```
correlation_plot(data=data,output_path='/content/')
```

```
Image saved at /content/Corr_plot.png
```



In [ ]:

```
# Find correlations with the target and sort
data_trans = data.copy()
data_trans['Target'] = data_trans['Curr Status'].apply(
    lambda x: 0 if x == 'No' else 1)
data_trans = data_trans.drop(
    ['Curr Status', 'EmployeeCount', 'EmployeeNumber', 'StandardHours', 'O
ver18'], axis=1)
correlations = data_trans.corr()['Target'].sort_values()
print('Most Positive Correlations: \n', correlations.tail(5))
print('\nMost Negative Correlations: \n', correlations.head(5))
```

```
Most Positive Correlations:
 PerformanceRating     0.002889
MonthlyRate            0.015170
NumCompaniesWorked     0.043494
DistanceFromHome       0.077924
Target                 1.000000
Name: Target, dtype: float64

Most Negative Correlations:
 TotalWorkingYears    -0.171063
JobLevel              -0.169105
YearsInCurrentRole    -0.160545
MonthlyIncome         -0.159840
Age                   -0.159205
Name: Target, dtype: float64
```

Heat Map

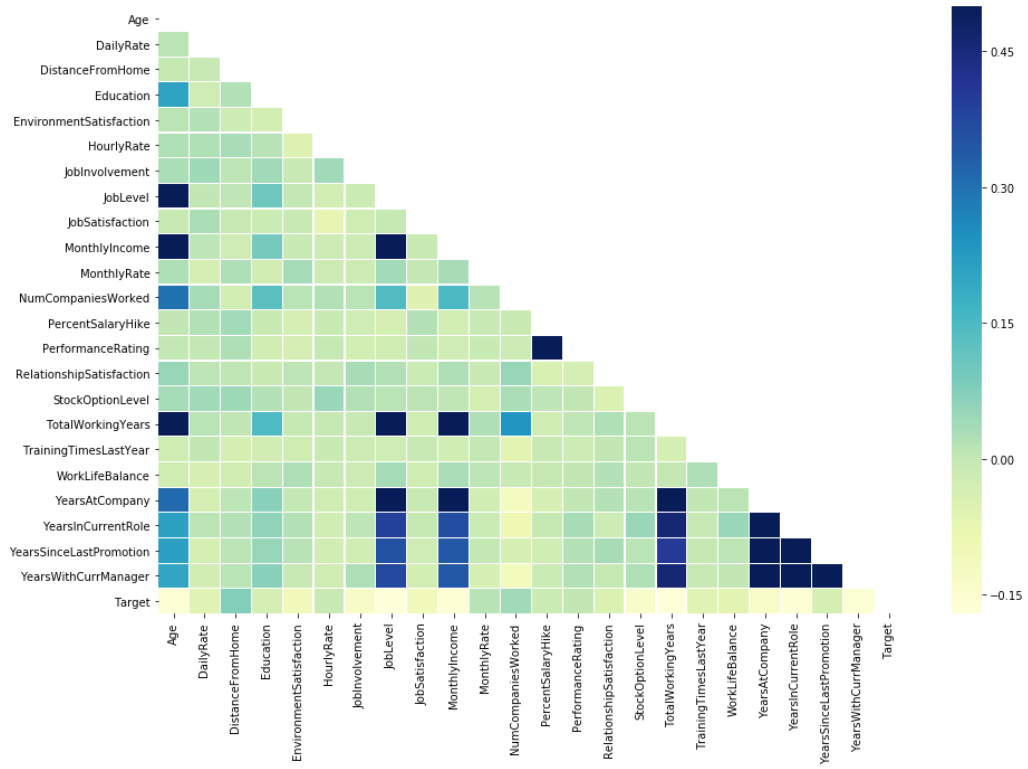
> Let's plot a heatmap to visualize the correlation between Attrition and these factors.

In [ ]:

```
# Calculate correlations
corr = data_trans.corr()
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)] = True
# Heatmap
plt.figure(figsize=(15, 10))
sns.heatmap(corr,
            vmax=.5,
            mask=mask,
            # annot=True, fmt='.2f',
            linewidths=.2, cmap="YlGnBu")
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x187940655f8>
```



EDA Obtained Results

Let's summarise the findings from this EDA:

> - The dataset does not feature any missing or erroneous data values, and all features are of the correct data type.

- The dataset is **imbalanced** with the majoriy of observations describing Currently Active Employees.

Pre-processing

Encoding

In [ ]:

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
# Create a label encoder object
le = LabelEncoder()
```

In [ ]:

```
print(data.shape)
data.head()
```

```
(1048, 35)
```

Out[ ]:

	Sr.No	Curr Status	Exit Date	Exit Type	Title	Gender	Designation	Department	S Departme
0	1	0	NaT	NaN	Ms.	Female	Senior Manager	Project Management	Proj Manageme
1	2	0	NaT	NaN	Mr.	Male	Senior Executive	FRD - Injectable	Formulatic Resear
2	3	0	NaT	NaN	Ms.	Female	Deputy Manager	RAD- II	Regulat Affairs
3	4	0	NaT	NaN	Ms.	Female	Senior Manager	Administration	Administrati
4	5	0	NaT	NaN	Mr.	Male	Principal Scientist- II	FARD	Formulatic Analyti Resear

In [ ]:

```
# Label Encoding will be used for columns with 2 or less unique values
le_count = 0
for col in data.columns[1:]:
    if data[col].dtype == 'object':
        if len(list(data[col].unique())) <= 2:
            le.fit(data[col])
            data[col] = le.transform(data[col])
            le_count += 1
print('{} columns were label encoded.'.format(le_count))
```

```
2 columns were label encoded.
```

In [ ]:

```
# convert rest of categorical variable into dummy
data = pd.get_dummies(data, drop_first=True)
```

In [ ]:

```
print(data.shape)
data.head()
```

```
(1048, 35)
```

Out[ ]:

	Sr.No	Curr Status	Exit Date	Exit Type	Title	Gender	Designation	Department	S Departme
0	1	Active	NaT	NaN	Ms.	Female	Senior Manager	Project Management	Proj Manageme
1	2	Active	NaT	NaN	Mr.	Male	Senior Executive	FRD - Injectable	Formulatic Resear
2	3	Active	NaT	NaN	Ms.	Female	Deputy Manager	RAD- II	Regulat Affairs
3	4	Active	NaT	NaN	Ms.	Female	Senior Manager	Administration	Administrati
4	5	Active	NaT	NaN	Mr.	Male	Principal Scientist- II	FARD	Formulatic Analyti Resear

Performing Feature Scaling

> Feature Scaling using MinMaxScaler, scaling between 0 and 5.

In [ ]:

```
# import MinMaxScaler
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 5))
HR_col = list(data.columns)
HR_col.remove('Curr Status')
for col in HR_col:
    data[col] = data[col].astype(float)
    data[[col]] = scaler.fit_transform(data[[col]])
data['Curr Status'] = pd.to_numeric(data['Curr Status'], downcast='float')
data.head()
```

In [ ]:

```
print('Size of Full Encoded Dataset: {}'. format(data.shape))
```

```
Size of Full Encoded Dataset: (1048, 35)
```

Splitting data into training and testing sets

> Prior to implementating or applying any Machine Learning algorithms, we must decouple training and testing datasets from our master dataframe.

In [ ]:

```
# assign the target to a new dataframe and convert it to a numerical featu
re
#df_target = data[['Curr Status']].copy()
target = data['Curr Status'].copy()
```

In [ ]:

```
type(target)
```

Out[ ]:

```
pandas.core.series.Series
```

In [ ]:

```
# let's remove the target feature and redundant features from the dataset
data.drop(['Curr Status', 'Blood Group', 'Car Allowance',
           'Employee Joining Type', 'FFC'], axis=1, inplace=True)
print('Size of Full dataset is: {}'.format(data.shape))
```

```
Size of Full dataset is: (1048, 30)
```

In [ ]:

```
# Since we have class imbalance (i.e. more employees with turnover=0 than
 turnover=1)
# let's use stratify=y to maintain the same ratio as in the training datas
et when splitting the dataset
X_train, X_test, y_train, y_test = train_test_split(data,
                                                    target,
                                                    test_size=0.25,
                                                    random_state=7,
                                                    stratify=target)
print("Number transactions X_train dataset: ", X_train.shape)
print("Number transactions y_train dataset: ", y_train.shape)
print("Number transactions X_test dataset: ", X_test.shape)
print("Number transactions y_test dataset: ", y_test.shape)
```

```
Number transactions X_train dataset:  (786, 30)
Number transactions y_train dataset:  (786,)

Number transactions X_test dataset:  (262, 30)
Number transactions y_test dataset:  (262,)
```

Using some Machine Learning Models

Baseline Algorithms

> Using a range of **baseline** algorithms and other algorithms like **Logistic Regression**, **Random Forest**, **SVM**, **KNN**, **Decision Tree Classifier**, **Gaussian NB**.

In [ ]:

```
# selection of algorithms to consider and set performance measure
models = []
models.append(('Logistic Regression', LogisticRegression(solver='liblinea
r', random_state=7,
                                                         class_weight='bal
anced')))
models.append(('Random Forest', RandomForestClassifier(
    n_estimators=100, random_state=7)))
models.append(('SVM', SVC(gamma='auto', random_state=7)))
models.append(('KNN', KNeighborsClassifier()))
models.append(('Decision Tree Classifier',
               DecisionTreeClassifier(random_state=7)))
models.append(('Gaussian NB', GaussianNB()))
```

> Let's evaluate each model in turn and provide accuracy and standard deviation scores

In [ ]:

```
from sklearn.model_selection import KFold

from sklearn import model_selection
import sklearn.model_selection
import sklearn
```

In [ ]:

```
acc_results = []
auc_results = []
names = []
# set table to table to populate with performance results
col = ['Algorithm', 'ROC AUC Mean', 'ROC AUC STD',
       'Accuracy Mean', 'Accuracy STD']
df_results = pd.DataFrame(columns=col)
i = 0
# evaluate each model using cross-validation
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, shuffle=True)  # 10-fold cr
oss-validation

    cv_acc_results = model_selection.cross_val_score(  # accuracy scoring
        model, X_train, y_train, cv=kfold, scoring='accuracy')

    cv_auc_results = model_selection.cross_val_score(  # roc_auc scoring
        model, X_train, y_train, cv=kfold, scoring='roc_auc')
```

```
    acc_results.append(cv_acc_results)
    auc_results.append(cv_auc_results)
    names.append(name)
    df_results.loc[i] = [name,
                         round(cv_auc_results.mean()*100, 2),
                         round(cv_auc_results.std()*100, 2),
                         round(cv_acc_results.mean()*100, 2),
                         round(cv_acc_results.std()*100, 2)
                         ]
    i += 1
df_results.sort_values(by=['ROC AUC Mean'], ascending=False)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/model_selectio
n/_validation.py:372: FitFailedWarning:


10 fits failed out of a total of 10.
The score on these train-test partitions for these parameters
will be set to nan.
If these failures are not expected, you can try to debug them
by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------
------------------
10 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/dist-packages/sklearn/model_
selection/_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/linear
_model/_logistic.py", line 1514, in fit
    accept_large_sparse=solver not in ["liblinear", "sag", "s
aga"],
  File "/usr/local/lib/python3.7/dist-packages/sklearn/base.p
y", line 581, in _validate_data
    X, y = check_X_y(X, y, **check_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/utils/
validation.py", line 976, in check_X_y
    estimator=estimator,
  File "/usr/local/lib/python3.7/dist-packages/sklearn/utils/
validation.py", line 665, in check_array
    dtype_orig = np.result_type(*dtypes_orig)
  File "<__array_function__ internals>", line 6, in result_ty
pe
TypeError: The DType <class 'numpy.dtype[float64]'> could not
be promoted by <class 'numpy.dtype[datetime64]'>. This means
that no common DType exists for the given inputs. For example
they cannot be stored in a single array unless the dtype is `
object`. The full list of DTypes is: (<class 'numpy.dtype[flo
at64]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype
[object_]'>, <class 'numpy.dtype[object_]'>, <class 'numpy.dt
ype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'nump
y.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'n
umpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class
'numpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <cla
ss 'numpy.dtype[object_]'>, <class 'numpy.dtype[datetime6
4]'>, <class 'numpy.dtype[datetime64]'>, <class 'numpy.dtype
[datetime64]'>, <class 'numpy.dtype[float64]'>, <class 'nump
y.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'n
```

```
y.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'n
umpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class
'numpy.dtype[object_]'>, <class 'numpy.dtype[float64]'>, <cla
ss 'numpy.dtype[object_]'>, <class 'numpy.dtype[float64]'>, <
class 'numpy.dtype[float64]'>, <class 'numpy.dtype[float6
4]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype[flo
at64]'>, <class 'numpy.dtype[int64]'>)


/usr/local/lib/python3.7/dist-packages/sklearn/model_selectio
n/_validation.py:372: FitFailedWarning:


10 fits failed out of a total of 10.
The score on these train-test partitions for these parameters
will be set to nan.
If these failures are not expected, you can try to debug them
by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------
-------------------
10 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/dist-packages/sklearn/model_
selection/_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/linear
_model/_logistic.py", line 1514, in fit
    accept_large_sparse=solver not in ["liblinear", "sag", "s
aga"],
  File "/usr/local/lib/python3.7/dist-packages/sklearn/base.p
y", line 581, in _validate_data
    X, y = check_X_y(X, y, **check_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/utils/
validation.py", line 976, in check_X_y
    estimator=estimator,
  File "/usr/local/lib/python3.7/dist-packages/sklearn/utils/
validation.py", line 665, in check_array
    dtype_orig = np.result_type(*dtypes_orig)
  File "<__array_function__ internals>", line 6, in result_ty
pe
TypeError: The DType <class 'numpy.dtype[float64]'> could not
be promoted by <class 'numpy.dtype[datetime64]'>. This means
that no common DType exists for the given inputs. For example
they cannot be stored in a single array unless the dtype is `
object`. The full list of DTypes is: (<class 'numpy.dtype[flo
at64]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype
[object_]'>, <class 'numpy.dtype[object_]'>, <class 'numpy.dt
ype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'nump
y.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'n
umpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class
'numpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <cla
ss 'numpy.dtype[object_]'>, <class 'numpy.dtype[datetime6
4]'>, <class 'numpy.dtype[datetime64]'>, <class 'numpy.dtype
[datetime64]'>, <class 'numpy.dtype[float64]'>, <class 'nump
y.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'n
umpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class
'numpy.dtype[object_]'>, <class 'numpy.dtype[float64]'>, <cla
ss 'numpy.dtype[object_]'>, <class 'numpy.dtype[float64]'>, <
class 'numpy.dtype[float64]'>, <class 'numpy.dtype[float6
```

```
4]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype[flo
at64]'>, <class 'numpy.dtype[int64]'>)


/usr/local/lib/python3.7/dist-packages/sklearn/model_selectio
n/_validation.py:372: FitFailedWarning:


10 fits failed out of a total of 10.
The score on these train-test partitions for these parameters
will be set to nan.
If these failures are not expected, you can try to debug them
by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------
------------------
10 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/dist-packages/sklearn/model_
selection/_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/ensemb
le/_forest.py", line 328, in fit
    X, y, multi_output=True, accept_sparse="csc", dtype=DTYPE
  File "/usr/local/lib/python3.7/dist-packages/sklearn/base.p
y", line 581, in _validate_data
    X, y = check_X_y(X, y, **check_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/utils/
validation.py", line 976, in check_X_y
    estimator=estimator,
  File "/usr/local/lib/python3.7/dist-packages/sklearn/utils/
validation.py", line 665, in check_array
    dtype_orig = np.result_type(*dtypes_orig)
  File "<__array_function__ internals>", line 6, in result_ty
pe
TypeError: The DType <class 'numpy.dtype[float64]'> could not
be promoted by <class 'numpy.dtype[datetime64]'>. This means
that no common DType exists for the given inputs. For example
they cannot be stored in a single array unless the dtype is `
object`. The full list of DTypes is: (<class 'numpy.dtype[flo
at64]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype
[object_]'>, <class 'numpy.dtype[object_]'>, <class 'numpy.dt
ype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'nump
y.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'n
umpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class
'numpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <cla
ss 'numpy.dtype[object_]'>, <class 'numpy.dtype[datetime6
4]'>, <class 'numpy.dtype[datetime64]'>, <class 'numpy.dtype
[datetime64]'>, <class 'numpy.dtype[float64]'>, <class 'nump
y.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'n
umpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class
'numpy.dtype[object_]'>, <class 'numpy.dtype[float64]'>, <cla
ss 'numpy.dtype[object_]'>, <class 'numpy.dtype[float64]'>, <
class 'numpy.dtype[float64]'>, <class 'numpy.dtype[float6
4]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype[flo
at64]'>, <class 'numpy.dtype[int64]'>)


/usr/local/lib/python3.7/dist-packages/sklearn/model_selectio
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/model_selectio
n/_validation.py:372: FitFailedWarning:


10 fits failed out of a total of 10.
The score on these train-test partitions for these parameters
will be set to nan.
If these failures are not expected, you can try to debug them
by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------
------------------
10 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/dist-packages/sklearn/model_
selection/_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/ensemb
le/_forest.py", line 328, in fit
    X, y, multi_output=True, accept_sparse="csc", dtype=DTYPE
  File "/usr/local/lib/python3.7/dist-packages/sklearn/base.p
y", line 581, in _validate_data
    X, y = check_X_y(X, y, **check_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/utils/
validation.py", line 976, in check_X_y
    estimator=estimator,
  File "/usr/local/lib/python3.7/dist-packages/sklearn/utils/
validation.py", line 665, in check_array
    dtype_orig = np.result_type(*dtypes_orig)
  File "<__array_function__ internals>", line 6, in result_ty
pe
TypeError: The DType <class 'numpy.dtype[float64]'> could not
be promoted by <class 'numpy.dtype[datetime64]'>. This means
that no common DType exists for the given inputs. For example
they cannot be stored in a single array unless the dtype is `
object`. The full list of DTypes is: (<class 'numpy.dtype[flo
at64]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype
[object_]'>, <class 'numpy.dtype[object_]'>, <class 'numpy.dt
ype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'nump
y.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'n
umpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class
'numpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <cla
ss 'numpy.dtype[object_]'>, <class 'numpy.dtype[datetime6
4]'>, <class 'numpy.dtype[datetime64]'>, <class 'numpy.dtype
[datetime64]'>, <class 'numpy.dtype[float64]'>, <class 'nump
y.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'n
umpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class
'numpy.dtype[object_]'>, <class 'numpy.dtype[float64]'>, <cla
ss 'numpy.dtype[object_]'>, <class 'numpy.dtype[float64]'>, <
class 'numpy.dtype[float64]'>, <class 'numpy.dtype[float6
4]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype[flo
at64]'>, <class 'numpy.dtype[int64]'>)


/usr/local/lib/python3.7/dist-packages/sklearn/model_selectio
n/_validation.py:372: FitFailedWarning:


10 fits failed out of a total of 10.
The score on these train-test partitions for these parameters
```

```
The score on these train-test partitions for these parameters
will be set to nan.
If these failures are not expected, you can try to debug them
by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------
-------------------
10 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/dist-packages/sklearn/model_
selection/_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/svm/_b
ase.py", line 196, in fit
    accept_large_sparse=False,
  File "/usr/local/lib/python3.7/dist-packages/sklearn/base.p
y", line 581, in _validate_data
    X, y = check_X_y(X, y, **check_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/utils/
validation.py", line 976, in check_X_y
    estimator=estimator,
  File "/usr/local/lib/python3.7/dist-packages/sklearn/utils/
validation.py", line 665, in check_array
    dtype_orig = np.result_type(*dtypes_orig)
  File "<__array_function__ internals>", line 6, in result_ty
pe
TypeError: The DType <class 'numpy.dtype[float64]'> could not
be promoted by <class 'numpy.dtype[datetime64]'>. This means
that no common DType exists for the given inputs. For example
they cannot be stored in a single array unless the dtype is `
object`. The full list of DTypes is: (<class 'numpy.dtype[flo
at64]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype
[object_]'>, <class 'numpy.dtype[object_]'>, <class 'numpy.dt
ype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'nump
y.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'n
umpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class
'numpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <cla
ss 'numpy.dtype[object_]'>, <class 'numpy.dtype[datetime6
4]'>, <class 'numpy.dtype[datetime64]'>, <class 'numpy.dtype
[datetime64]'>, <class 'numpy.dtype[float64]'>, <class 'nump
y.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'n
umpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class
'numpy.dtype[object_]'>, <class 'numpy.dtype[float64]'>, <cla
ss 'numpy.dtype[object_]'>, <class 'numpy.dtype[float64]'>, <
class 'numpy.dtype[float64]'>, <class 'numpy.dtype[float6
4]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype[flo
at64]'>, <class 'numpy.dtype[int64]'>)


/usr/local/lib/python3.7/dist-packages/sklearn/model_selectio
n/_validation.py:372: FitFailedWarning:


10 fits failed out of a total of 10.
The score on these train-test partitions for these parameters
will be set to nan.
If these failures are not expected, you can try to debug them
by setting error_score='raise'.

Below are more details about the failures:
```

```
Below are more details about the failures:
--------------------------------------------------------------------------------
10 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py", line 196, in fit
    accept_large_sparse=False,
  File "/usr/local/lib/python3.7/dist-packages/sklearn/base.py", line 581, in _validate_data
    X, y = check_X_y(X, y, **check_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py", line 976, in check_X_y
    estimator=estimator,
  File "/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py", line 665, in check_array
    dtype_orig = np.result_type(*dtypes_orig)
  File "<__array_function__ internals>", line 6, in result_type
TypeError: The DType <class 'numpy.dtype[float64]'> could not be promoted by <class 'numpy.dtype[datetime64]'>. This means that no common DType exists for the given inputs. For example they cannot be stored in a single array unless the dtype is `object`. The full list of DTypes is: (<class 'numpy.dtype[float64]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'numpy.dtype[datetime64]'>, <class 'numpy.dtype[datetime64]'>, <class 'numpy.dtype[datetime64]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype[object_]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype[int64]'>)


/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:372: FitFailedWarning:


10 fits failed out of a total of 10.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
10 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/dist-packages/sklearn/model
```

```
  File "/usr/local/lib/python3.7/dist-packages/sklearn/model_
selection/_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/neighb
ors/_classification.py", line 198, in fit
    return self._fit(X, y)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/neighb
ors/_base.py", line 400, in _fit
    X, y = self._validate_data(X, y, accept_sparse="csr", mul
ti_output=True)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/base.p
y", line 581, in _validate_data
    X, y = check_X_y(X, y, **check_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/utils/
validation.py", line 976, in check_X_y
    estimator=estimator,
  File "/usr/local/lib/python3.7/dist-packages/sklearn/utils/
validation.py", line 665, in check_array
    dtype_orig = np.result_type(*dtypes_orig)
  File "<__array_function__ internals>", line 6, in result_ty
pe
TypeError: The DType <class 'numpy.dtype[float64]'> could not
be promoted by <class 'numpy.dtype[datetime64]'>. This means
that no common DType exists for the given inputs. For example
they cannot be stored in a single array unless the dtype is `
object`. The full list of DTypes is: (<class 'numpy.dtype[flo
at64]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype
[object_]'>, <class 'numpy.dtype[object_]'>, <class 'numpy.dt
ype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'nump
y.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'n
umpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class
'numpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <cla
ss 'numpy.dtype[object_]'>, <class 'numpy.dtype[datetime6
4]'>, <class 'numpy.dtype[datetime64]'>, <class 'numpy.dtype
[datetime64]'>, <class 'numpy.dtype[float64]'>, <class 'nump
y.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'n
umpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class
'numpy.dtype[object_]'>, <class 'numpy.dtype[float64]'>, <cla
ss 'numpy.dtype[object_]'>, <class 'numpy.dtype[float64]'>, <
class 'numpy.dtype[float64]'>, <class 'numpy.dtype[float6
4]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype[flo
at64]'>, <class 'numpy.dtype[int64]'>)


/usr/local/lib/python3.7/dist-packages/sklearn/model_selectio
n/_validation.py:372: FitFailedWarning:


10 fits failed out of a total of 10.
The score on these train-test partitions for these parameters
will be set to nan.
If these failures are not expected, you can try to debug them
by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------
------------------
10 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/dist-packages/sklearn/model_
selection/_validation.py", line 680, in _fit_and_score
```

```
selection/_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/neighb
ors/_classification.py", line 198, in fit
    return self._fit(X, y)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/neighb
ors/_base.py", line 400, in _fit
    X, y = self._validate_data(X, y, accept_sparse="csr", mul
ti_output=True)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/base.p
y", line 581, in _validate_data
    X, y = check_X_y(X, y, **check_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/utils/
validation.py", line 976, in check_X_y
    estimator=estimator,
  File "/usr/local/lib/python3.7/dist-packages/sklearn/utils/
validation.py", line 665, in check_array
    dtype_orig = np.result_type(*dtypes_orig)
  File "<__array_function__ internals>", line 6, in result_ty
pe
TypeError: The DType <class 'numpy.dtype[float64]'> could not
be promoted by <class 'numpy.dtype[datetime64]'>. This means
that no common DType exists for the given inputs. For example
they cannot be stored in a single array unless the dtype is `
object`. The full list of DTypes is: (<class 'numpy.dtype[flo
at64]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype
[object_]'>, <class 'numpy.dtype[object_]'>, <class 'numpy.dt
ype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'nump
y.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'n
umpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class
'numpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <cla
ss 'numpy.dtype[object_]'>, <class 'numpy.dtype[datetime6
4]'>, <class 'numpy.dtype[datetime64]'>, <class 'numpy.dtype
[datetime64]'>, <class 'numpy.dtype[float64]'>, <class 'nump
y.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'n
umpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class
'numpy.dtype[object_]'>, <class 'numpy.dtype[float64]'>, <cla
ss 'numpy.dtype[object_]'>, <class 'numpy.dtype[float64]'>, <
class 'numpy.dtype[float64]'>, <class 'numpy.dtype[float6
4]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype[flo
at64]'>, <class 'numpy.dtype[int64]'>)


/usr/local/lib/python3.7/dist-packages/sklearn/model_selectio
n/_validation.py:372: FitFailedWarning:


10 fits failed out of a total of 10.
The score on these train-test partitions for these parameters
will be set to nan.
If these failures are not expected, you can try to debug them
by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------
------------------
10 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/dist-packages/sklearn/model_
selection/_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
```

```
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/tree/_
classes.py", line 942, in fit
    X_idx_sorted=X_idx_sorted,
  File "/usr/local/lib/python3.7/dist-packages/sklearn/tree/_
classes.py", line 166, in fit
    X, y, validate_separately=(check_X_params, check_y_param
s)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/base.p
y", line 578, in _validate_data
    X = check_array(X, **check_X_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/utils/
validation.py", line 665, in check_array
    dtype_orig = np.result_type(*dtypes_orig)
  File "<__array_function__ internals>", line 6, in result_ty
pe
TypeError: The DType <class 'numpy.dtype[float64]'> could not
be promoted by <class 'numpy.dtype[datetime64]'>. This means
that no common DType exists for the given inputs. For example
they cannot be stored in a single array unless the dtype is `
object`. The full list of DTypes is: (<class 'numpy.dtype[flo
at64]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype
[object_]'>, <class 'numpy.dtype[object_]'>, <class 'numpy.dt
ype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'nump
y.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'n
umpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class
'numpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <cla
ss 'numpy.dtype[object_]'>, <class 'numpy.dtype[datetime6
4]'>, <class 'numpy.dtype[datetime64]'>, <class 'numpy.dtype
[datetime64]'>, <class 'numpy.dtype[float64]'>, <class 'nump
y.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'n
umpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class
'numpy.dtype[object_]'>, <class 'numpy.dtype[float64]'>, <cla
ss 'numpy.dtype[object_]'>, <class 'numpy.dtype[float64]'>, <
class 'numpy.dtype[float64]'>, <class 'numpy.dtype[float6
4]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype[flo
at64]'>, <class 'numpy.dtype[int64]'>)


/usr/local/lib/python3.7/dist-packages/sklearn/model_selectio
n/_validation.py:372: FitFailedWarning:


10 fits failed out of a total of 10.
The score on these train-test partitions for these parameters
will be set to nan.
If these failures are not expected, you can try to debug them
by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------
------------------
10 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/dist-packages/sklearn/model_
selection/_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/tree/_
classes.py", line 942, in fit
    X_idx_sorted=X_idx_sorted,
  File "/usr/local/lib/python3.7/dist-packages/sklearn/tree/
```

```
  File "/usr/local/lib/python3.7/dist-packages/sklearn/tree/_
classes.py", line 166, in fit
    X, y, validate_separately=(check_X_params, check_y_param
s)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/base.p
y", line 578, in _validate_data
    X = check_array(X, **check_X_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/utils/
validation.py", line 665, in check_array
    dtype_orig = np.result_type(*dtypes_orig)
  File "<__array_function__ internals>", line 6, in result_ty
pe
TypeError: The DType <class 'numpy.dtype[float64]'> could not
be promoted by <class 'numpy.dtype[datetime64]'>. This means
that no common DType exists for the given inputs. For example
they cannot be stored in a single array unless the dtype is `
object`. The full list of DTypes is: (<class 'numpy.dtype[flo
at64]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype
[object_]'>, <class 'numpy.dtype[object_]'>, <class 'numpy.dt
ype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'nump
y.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'n
umpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class
'numpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <cla
ss 'numpy.dtype[object_]'>, <class 'numpy.dtype[datetime6
4]'>, <class 'numpy.dtype[datetime64]'>, <class 'numpy.dtype
[datetime64]'>, <class 'numpy.dtype[float64]'>, <class 'nump
y.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'n
umpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class
'numpy.dtype[object_]'>, <class 'numpy.dtype[float64]'>, <cla
ss 'numpy.dtype[object_]'>, <class 'numpy.dtype[float64]'>, <
class 'numpy.dtype[float64]'>, <class 'numpy.dtype[float6
4]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype[flo
at64]'>, <class 'numpy.dtype[int64]'>)


/usr/local/lib/python3.7/dist-packages/sklearn/model_selectio
n/_validation.py:372: FitFailedWarning:


10 fits failed out of a total of 10.
The score on these train-test partitions for these parameters
will be set to nan.
If these failures are not expected, you can try to debug them
by setting error_score='raise'.

Below are more details about the failures:
----------------------------------------------------------------
--------------------
10 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/dist-packages/sklearn/model_
selection/_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/naive_
bayes.py", line 246, in fit
    X, y, np.unique(y), _refit=True, sample_weight=sample_wei
ght
  File "/usr/local/lib/python3.7/dist-packages/sklearn/naive_
bayes.py", line 402, in _partial_fit
    X, y = self._validate_data(X, y, reset=first_call)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/base.p
```

```
  File "/usr/local/lib/python3.7/dist-packages/sklearn/base.p
y", line 581, in _validate_data
    X, y = check_X_y(X, y, **check_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/utils/
validation.py", line 976, in check_X_y
    estimator=estimator,
  File "/usr/local/lib/python3.7/dist-packages/sklearn/utils/
validation.py", line 665, in check_array
    dtype_orig = np.result_type(*dtypes_orig)
  File "<__array_function__ internals>", line 6, in result_ty
pe
TypeError: The DType <class 'numpy.dtype[float64]'> could not
be promoted by <class 'numpy.dtype[datetime64]'>. This means
that no common DType exists for the given inputs. For example
they cannot be stored in a single array unless the dtype is `
object`. The full list of DTypes is: (<class 'numpy.dtype[flo
at64]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype
[object_]'>, <class 'numpy.dtype[object_]'>, <class 'numpy.dt
ype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'nump
y.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'n
umpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class
'numpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <cla
ss 'numpy.dtype[object_]'>, <class 'numpy.dtype[datetime6
4]'>, <class 'numpy.dtype[datetime64]'>, <class 'numpy.dtype
[datetime64]'>, <class 'numpy.dtype[float64]'>, <class 'nump
y.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'n
umpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class
'numpy.dtype[object_]'>, <class 'numpy.dtype[float64]'>, <cla
ss 'numpy.dtype[object_]'>, <class 'numpy.dtype[float64]'>, <
class 'numpy.dtype[float64]'>, <class 'numpy.dtype[float6
4]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype[flo
at64]'>, <class 'numpy.dtype[int64]'>)


/usr/local/lib/python3.7/dist-packages/sklearn/model_selectio
n/_validation.py:372: FitFailedWarning:


10 fits failed out of a total of 10.
The score on these train-test partitions for these parameters
will be set to nan.
If these failures are not expected, you can try to debug them
by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------
--------------------
10 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/dist-packages/sklearn/model_
selection/_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/naive_
bayes.py", line 246, in fit
    X, y, np.unique(y), _refit=True, sample_weight=sample_wei
ght
  File "/usr/local/lib/python3.7/dist-packages/sklearn/naive_
bayes.py", line 402, in _partial_fit
    X, y = self._validate_data(X, y, reset=first_call)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/base.p
y", line 581, in _validate_data
```

```
y , line 961, in _validate_data
    X, y = check_X_y(X, y, **check_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/utils/
validation.py", line 976, in check_X_y
    estimator=estimator,
  File "/usr/local/lib/python3.7/dist-packages/sklearn/utils/
validation.py", line 665, in check_array
    dtype_orig = np.result_type(*dtypes_orig)
  File "<__array_function__ internals>", line 6, in result_ty
pe
TypeError: The DType <class 'numpy.dtype[float64]'> could not
be promoted by <class 'numpy.dtype[datetime64]'>. This means
that no common DType exists for the given inputs. For example
they cannot be stored in a single array unless the dtype is `
object`. The full list of DTypes is: (<class 'numpy.dtype[flo
at64]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype
[object_]'>, <class 'numpy.dtype[object_]'>, <class 'numpy.dt
ype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'nump
y.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'n
umpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class
'numpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <cla
ss 'numpy.dtype[object_]'>, <class 'numpy.dtype[datetime6
4]'>, <class 'numpy.dtype[datetime64]'>, <class 'numpy.dtype
[datetime64]'>, <class 'numpy.dtype[float64]'>, <class 'nump
y.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class 'n
umpy.dtype[object_]'>, <class 'numpy.dtype[object_]'>, <class
'numpy.dtype[object_]'>, <class 'numpy.dtype[float64]'>, <cla
ss 'numpy.dtype[object_]'>, <class 'numpy.dtype[float64]'>, <
class 'numpy.dtype[float64]'>, <class 'numpy.dtype[float6
4]'>, <class 'numpy.dtype[float64]'>, <class 'numpy.dtype[flo
at64]'>, <class 'numpy.dtype[int64]'>)
```

Out[ ]:

	Algorithm	ROC AUC Mean	ROC AUC STD	Accuracy Mean	Accuracy STD
0	Logistic Regression	NaN	NaN	NaN	NaN
1	Random Forest	NaN	NaN	NaN	NaN
2	SVM	NaN	NaN	NaN	NaN
3	KNN	NaN	NaN	NaN	NaN
4	Decision Tree Classifier	NaN	NaN	NaN	NaN
5	Gaussian NB	NaN	NaN	NaN	NaN

> **Classification Accuracy**

In [ ]:

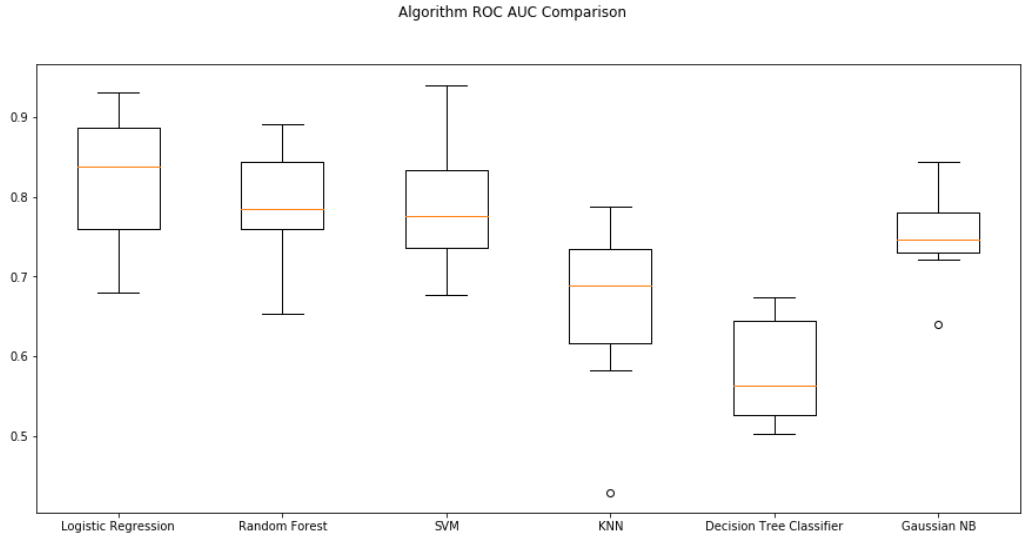
```
fig = plt.figure(figsize=(15, 7))
fig.suptitle('Algorithm Accuracy Comparison')
ax = fig.add_subplot(111)
```

```
plt.boxplot(acc_results)
ax.set_xticklabels(names)
plt.show()
```

> **Area under ROC Curve**

In [ ]:

```
fig = plt.figure(figsize=(15, 7))
fig.suptitle('Algorithm ROC AUC Comparison')
ax = fig.add_subplot(111)
plt.boxplot(auc_results)
ax.set_xticklabels(names)
plt.show()
```



> Based on our ROC AUC comparison analysis, **Logistic Regression** and **Random Forest** show the highest mean AUC scores. We will shortlist these two algorithms for further analysis. See below for more details on these two algos.

Logistic Regression

Logistic Regression

> Let's take a closer look at using the Logistic Regression algorithm. I'll be using 10 fold Cross-Validation to train our Logistic Regression Model and estimate its AUC score.

In [ ]:

```
kfold = model_selection.KFold(n_splits=10, random_state=7)
modelCV = LogisticRegression(solver='liblinear',
```

```
                                class_weight="balanced",
                                random_state=7)
scoring = 'roc_auc'
results = model_selection.cross_val_score(
    modelCV, X_train, y_train, cv=kfold, scoring=scoring)
print("AUC score (STD): %.2f (%.2f)" % (results.mean(), results.std()))
```

```
AUC score (STD): 0.82 (0.08)
```

Fine-tuning

> GridSearchCV allows use to fine-tune hyper-parameters by searching over specified parameter values for an estimator.

In [ ]:

```
param_grid = {'C': np.arange(1e-03, 2, 0.01)} # hyper-parameter list to fi
ne-tune
log_gs = GridSearchCV(LogisticRegression(solver='liblinear', # setting Gri
dSearchCV
                                         class_weight="balanced",
                                         random_state=7),
                      iid=True,
                      return_train_score=True,
                      param_grid=param_grid,
                      scoring='roc_auc',
                      cv=10)

log_grid = log_gs.fit(X_train, y_train)
log_opt = log_grid.best_estimator_
results = log_gs.cv_results_

print('='*20)
print("best params: " + str(log_gs.best_estimator_))
print("best params: " + str(log_gs.best_params_))
print('best score:', log_gs.best_score_)
print('='*20)
```

```
====================
best params: LogisticRegression(C=0.05099999999999999, class_
weight='balanced', dual=False,
          fit_intercept=True, intercept_scaling=1, max_iter=1
00,
          multi_class='warn', n_jobs=None, penalty='l2', rand
om_state=7,
          solver='liblinear', tol=0.0001, verbose=0, warm_sta
rt=False)
best params: {'C': 0.05099999999999999}
best score: 0.8180815631000706
====================
```

> As shown above, the results from GridSearchCV provided us with fine-tuned hyper-parameter using ROC_AUC as the scoring metric.

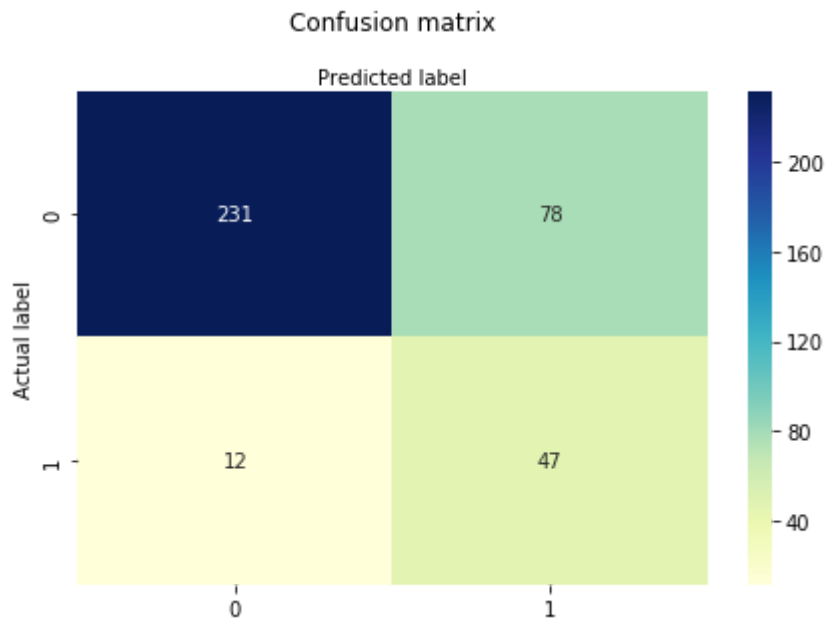
Evaluation

In [ ]:

```
## Confusion Matrix
cnf_matrix = metrics.confusion_matrix(y_test, log_opt.predict(X_test))
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

Out[ ]:

```
Text(0.5,257.44,'Predicted label')
```



In [ ]:

```
print('Accuracy of Logistic Regression Classifier on test set: {:.2f}'.for
mat(log_opt.score(X_test, y_test)*100))
```

```
Accuracy of Logistic Regression Classifier on test set: 75.54
```

> The Confusion matrix is telling us that we have 231+47 correct predictions and 78+12 incorrect predictions. In other words, an accurac of 75.54%.

In [ ]:

```
# Classification report for the optimised Log Regression
log_opt.fit(X_train, y_train)
print(classification_report(y_test, log_opt.predict(X_test)))
```

```
              precision    recall  f1-score   support

         0.0       0.95      0.75      0.84       309
         1.0       0.38      0.80      0.51        59

   micro avg       0.76      0.76      0.76       368
   macro avg       0.66      0.77      0.67       368
weighted avg       0.86      0.76      0.78       368
```

> The resulting AUC score is: 0.857 which is higher than that best score during the optimisation step.

In [ ]:

```
log_opt.fit(X_train, y_train) # fit optimised model to the training data
probs = log_opt.predict_proba(X_test) # predict probabilities
probs = probs[:, 1] # we will only keep probabilities associated with the
 employee leaving
logit_roc_auc = roc_auc_score(y_test, probs) # calculate AUC score using t
est dataset
print('AUC score: %.3f' % logit_roc_auc)
```

```
AUC score: 0.857
```

Random Forest Classifier

Fine-tuning

In [ ]:

```
rf_classifier = RandomForestClassifier(class_weight = "balanced",
                                       random_state=7)
param_grid = {'n_estimators': [50, 75, 100, 125, 150, 175],
              'min_samples_split':[2,4,6,8,10],
              'min_samples_leaf': [1, 2, 3, 4],
              'max_depth': [5, 10, 15, 20, 25]}

grid_obj = GridSearchCV(rf_classifier,
                        iid=True,
                        return_train_score=True,
                        param_grid=param_grid,
                        scoring='roc_auc',
                        cv=10)

grid_fit = grid_obj.fit(X_train, y_train)
rf_opt = grid_fit.best_estimator_

print('='*20)
print("best params: " + str(grid_obj.best_estimator_))
```

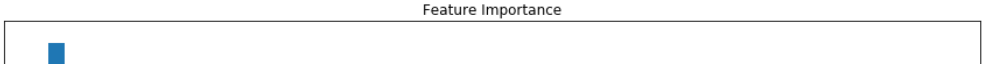
```
print("best params: " + str(grid_obj.best_params_))
print('best score:', grid_obj.best_score_)
print('='*20)
```

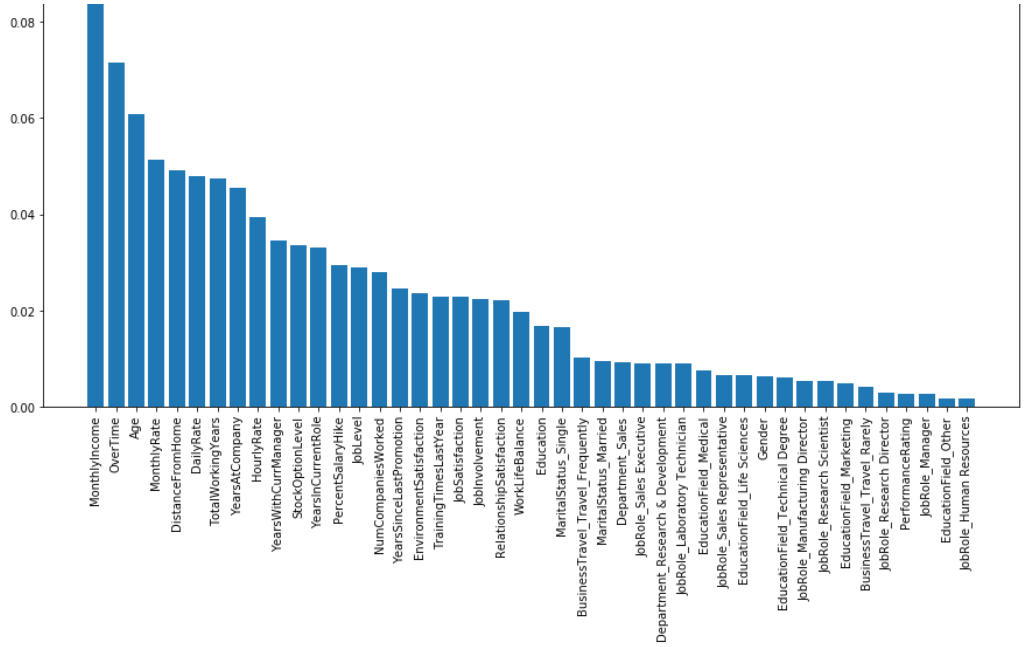
```
====================
best params: RandomForestClassifier(bootstrap=True, class_wei
ght='balanced',
            criterion='gini', max_depth=15, max_features='aut
o',
            max_leaf_nodes=None, min_impurity_decrease=0.0,
            min_impurity_split=None, min_samples_leaf=1,
            min_samples_split=8, min_weight_fraction_leaf=0.
0,
            n_estimators=75, n_jobs=None, oob_score=False, ra
ndom_state=7,
            verbose=0, warm_start=False)
best params: {'max_depth': 15, 'min_samples_leaf': 1, 'min_sa
mples_split': 8, 'n_estimators': 75}
best score: 0.7956083711198764
====================
```

> Random Forest allows us to know which features are of the most importance in predicting the target feature ("attrition" in this project). Below, we plot features by their importance.

In [ ]:

```
importances = rf_opt.feature_importances_
indices = np.argsort(importances)[::-1] # Sort feature importances in desc
ending order
names = [X_train.columns[i] for i in indices] # Rearrange feature names so
they match the sorted feature importances
plt.figure(figsize=(15, 7)) # Create plot
plt.title("Feature Importance") # Create plot title
plt.bar(range(X_train.shape[1]), importances[indices]) # Add bars
plt.xticks(range(X_train.shape[1]), names, rotation=90) # Add feature name
s as x-axis labels
plt.show() # Show plot
```





> The Top 10 most important indicators (ranked in the table below) as: MonthlyIncome, OverTime, Age, MonthlyRate, DistanceFromHome, DailyRate, TotalWorkingYears, YearsAtCompany, HourlyRate, YearsWithCurrManager.

In [ ]:

```
importances = rf_opt.feature_importances_
df_param_coeff = pd.DataFrame(columns=['Feature', 'Coefficient'])
for i in range(44):
    feat = X_train.columns[i]
    coeff = importances[i]
    df_param_coeff.loc[i] = (feat, coeff)
df_param_coeff.sort_values(by='Coefficient', ascending=False, inplace=True
)
df_param_coeff = df_param_coeff.reset_index(drop=True)
df_param_coeff.head(10)
```

Out[ ]:

	Feature	Coefficient
0	MonthlyIncome	0.088179
1	OverTime	0.071542
2	Age	0.060858
3	MonthlyRate	0.051225
4	DistanceFromHome	0.049128
5	DailyRate	0.047802
6	TotalWorkingYears	0.047392
7	YearsAtCompany	0.045513
8	HourlyRate	0.039374
9	YearsWithCurrManager	0.034526

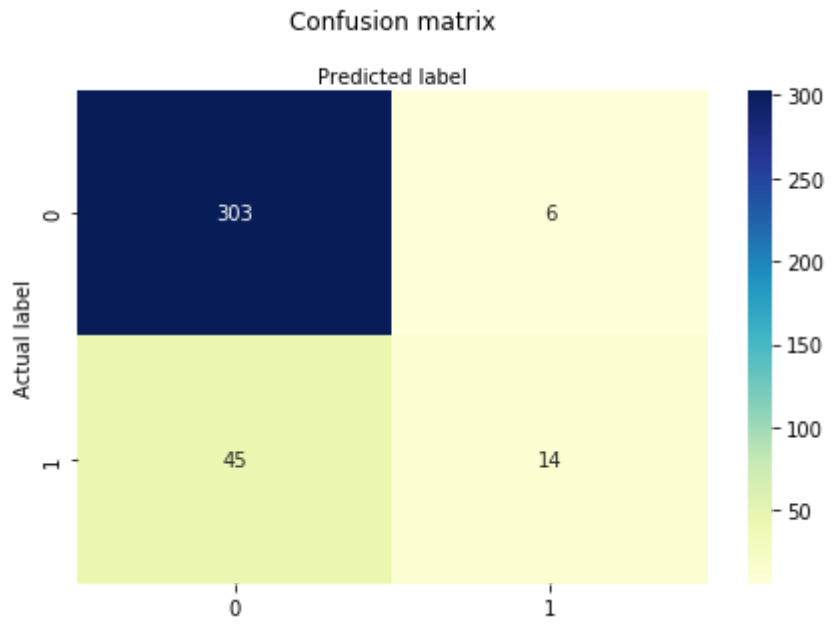
Evaluation

In [ ]:

```
## Confusion Matrix
cnf_matrix = metrics.confusion_matrix(y_test, rf_opt.predict(X_test))
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

Out[ ]:

```
Text(0.5,257.44,'Predicted label')
```



In [ ]:

```
print('Accuracy of RandomForest Regression Classifier on test set: {:.2f}'
.format(rf_opt.score(X_test, y_test)*100))
```

```
Accuracy of RandomForest Regression Classifier on test set: 8
6.14
```

> The Confusion matrix is telling us that we have 303+14 correct predictions and 1+52 incorrect predictions. In other words, an accurac of 86.14%.

In [ ]:

```
# Classification report for the optimised RF Regression
rf_opt.fit(X_train, y_train)
print(classification_report(y_test, rf_opt.predict(X_test)))
```

```
              precision    recall  f1-score   support

         0.0       0.87      0.98      0.92       309
         1.0       0.70      0.24      0.35        59

   micro avg       0.86      0.86      0.86       368
   macro avg       0.79      0.61      0.64       368
weighted avg       0.84      0.86      0.83       368
```

> Using probabilistic class prediction resulted in AUC score is: 0.818 which is higher than that best score during the optimisation step.

In [ ]:

```
rf_opt.fit(X_train, y_train) # fit optimised model to the training data
probs = rf_opt.predict_proba(X_test) # predict probabilities
probs = probs[:, 1] # we will only keep probabilities associated with the
 employee leaving
rf_opt_roc_auc = roc_auc_score(y_test, probs) # calculate AUC score using
 test dataset
print('AUC score: %.3f' % rf_opt_roc_auc)
```

```
AUC score: 0.818
```

ROC Graphs

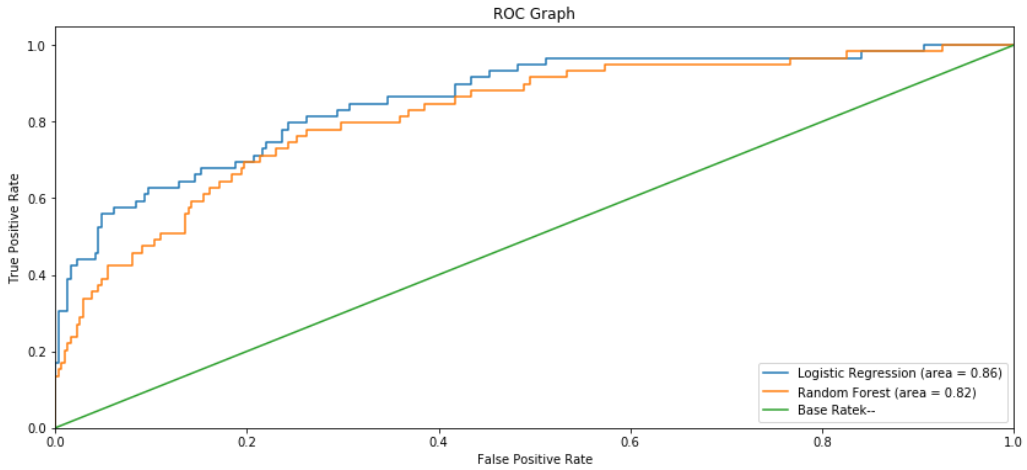
In [ ]:

```
# Create ROC Graph
from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(y_test, log_opt.predict_proba(X_test)[:,1
])
rf_fpr, rf_tpr, rf_thresholds = roc_curve(y_test, rf_opt.predict_proba(X_t
est)[:,1])
plt.figure(figsize=(14, 6))

# Plot Logistic Regression ROC
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_
auc)
# Plot Random Forest ROC
plt.plot(rf_fpr, rf_tpr, label='Random Forest (area = %0.2f)' % rf_opt_roc
_auc)
# Plot Base Rate ROC
plt.plot([0,1], [0,1],label='Base Rate' 'k--')

plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Graph')
```

```
plt.legend(loc="lower right")
plt.show()
```



> As shown below, the fine-tuned Logistic Regression model showed a higher AUC score compared to the Random Forest Classifier.

Conclusion Obtained from Above Predictions

> - Single employees show the largest proportion of leavers, compared to Married and Divorced counterparts.
> - Loyal employees with higher salaries and more responsbilities show lower proportion of leavers compared to their counterparts.
> - People who live further away from their work show higher proportion of leavers compared to their counterparts.
> - People who have less gross salaries show higher proportion of leavers compared to the less gross salaries people.
> - Employees that have experience at several companies previously show higher proportion of leavers compared new joinee.